AD-A102 493 PRC DATA SERVICES CO RESTON VA F/G 9/2
CALCOMP PREVIEW SYSTEM. PROGRAM MAINTENANCE MANUAL.(U)
APR 81 DCA100-77-C-0037
UNCLASSIFIED DCA/DF-81/002A NL

1 OF 2
AD A
102493

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. 18 DCA/DF-81/002a	2. AD-A102493	3. Recipient's Accession No.
4. Title and Subtitle CALCOMP PREVIEW SYSTEM, Program Maintenance Manual			5. Report Date April 1981 (date of issue)
			6.
7. Author(s) Planning Research Corporation (PRC) - Contractor			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Planning Research Corporation Data Services Company 11301 Sunset Hills Drive Reston, VA 22090			10. Project/Task/Work Unit No. Subtask 80-3.1
			11. Contract(C) or Grant(G) No. (C) DCA 100-77-C-0037 (G)
12. Sponsoring Organization Name and Address Defense Communications Engineering Center 1860 Wiehle Avenue Reston, VA 22090			13. Type of Report & Period Covered Final rept.
			14.
15. Supplementary Notes For magnetic tape see AD-A102492			

16. Abstract (Limit: 200 words)

The CALCOMP Preview System enables a user to evaluate the contents of a CALCOMP generated plot file by displaying a similar image on the Tektronix 4014 graphic display terminal.

AD A102493

17. Document Analysis a. Descriptors

b. Identifiers/Open-Ended Terms

A

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	Unclassified	
	20. Security Class (This Page) Unclassified	22. Price

DTIC FILE COPY

Control No. 80-3.1-2

DCA/DF-81-002A

PROGRAM MAINTENANCE MANUAL
for the
CALCOMP PREVIEW SYSTEM
Prepared for the
DEFENSE COMMUNICATIONS ENGINEERING CENTER
Contract No. DCA 100-77-C-0037
Subtask 80-3.1
April 1981

81 8 03 007

PRC (DCEC)
Control No. 80-3.1-2

PROGRAM MAINTENANCE MANUAL
for the
CALCOMP PREVIEW SYSTEM
Prepared for the
DEFENSE COMMUNICATIONS ENGINEERING CENTER
Contract No. DCA 100-77-C-0037
Subtask 80-3.1
April 1981

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

SECTION 1. GENERAL

1.1 Purpose of the Program Maintenance Manual. The objective of the Program Maintenance Manual for the CALCOMP Preview System (CPS) is to provide the maintenance programmer personnel with the information necessary to maintain the system.

1.2 Project References. The development of the CALCOMP Preview System was authorized under Subtask 80-3.1, CALCOMP to Tektronix Conversion, under contract DCA 100-77-C-0037. The following documentation provides information useful in understanding the system:

a. CALCOMP Model 7000 PLOTTING SYSTEM User's Manual, DCEC (R830), 1 Oct 76.

b. CALCOMP Basic Software, CAL EDIT User's guide, California Computer Products, Inc. May 1975.

c. TEKTRONIX PLOT 10 Terminal Control System User's Manual.

d. System/360 Scientific Subroutine Package (360A-CM-03X) Version II Application Description, 1967.

e. Users Manual for the CALCOMP Preview System, March 1981.

SECTION 2. SYSTEM DESCRIPTION

2.1 System Application. The CPS enables a user to evaluate the contents of a data tape prepared for the CALCOMP flatbed plotter by displaying a similar image on the Tektronix graphic display terminal. Representative uses include:

- Evaluating a plot file before plotting it on the CALCOMP plotter.
- Determining an appropriate scaling factor and offset before submitting the plot file to the CALCOMP plotter.
- Obtaining plot images when the CALCOMP plotter is inoperative or heavily used.

CPS provides a set of commands to control scaling, rotation, and positioning of the display on the terminal screen. At any time, the user may make a copy of the image using the hard copy attachment to the display terminal. This capability allows the user to access the data on a CALCOMP plot tape at will, without requiring the support of a plotter operator.

This preliminary review feature enables the user to confirm that the data is ready for plotting, and thus provides a significant saving in time and costs.

2.2 Security and Privacy. The CPS is unclassified, and is subject to no privacy restrictions. Each user is responsible for observing security or privacy restrictions applicable to any files plotted through the system.

2.3 General Description. The system consists of one program, which is run as a stand alone job when required. It is available for the HSF NASC AS/5-3 and on the MIGS PDP-11/70 system. Under the NASC system, a CALCOMP plot file must be placed on disk to be accessed under TSO. Under the PDP-11/70, the plot file may be either on disk or on a magnetic tape file. See Figures 2-01 and 2-02 for system flowcharts.

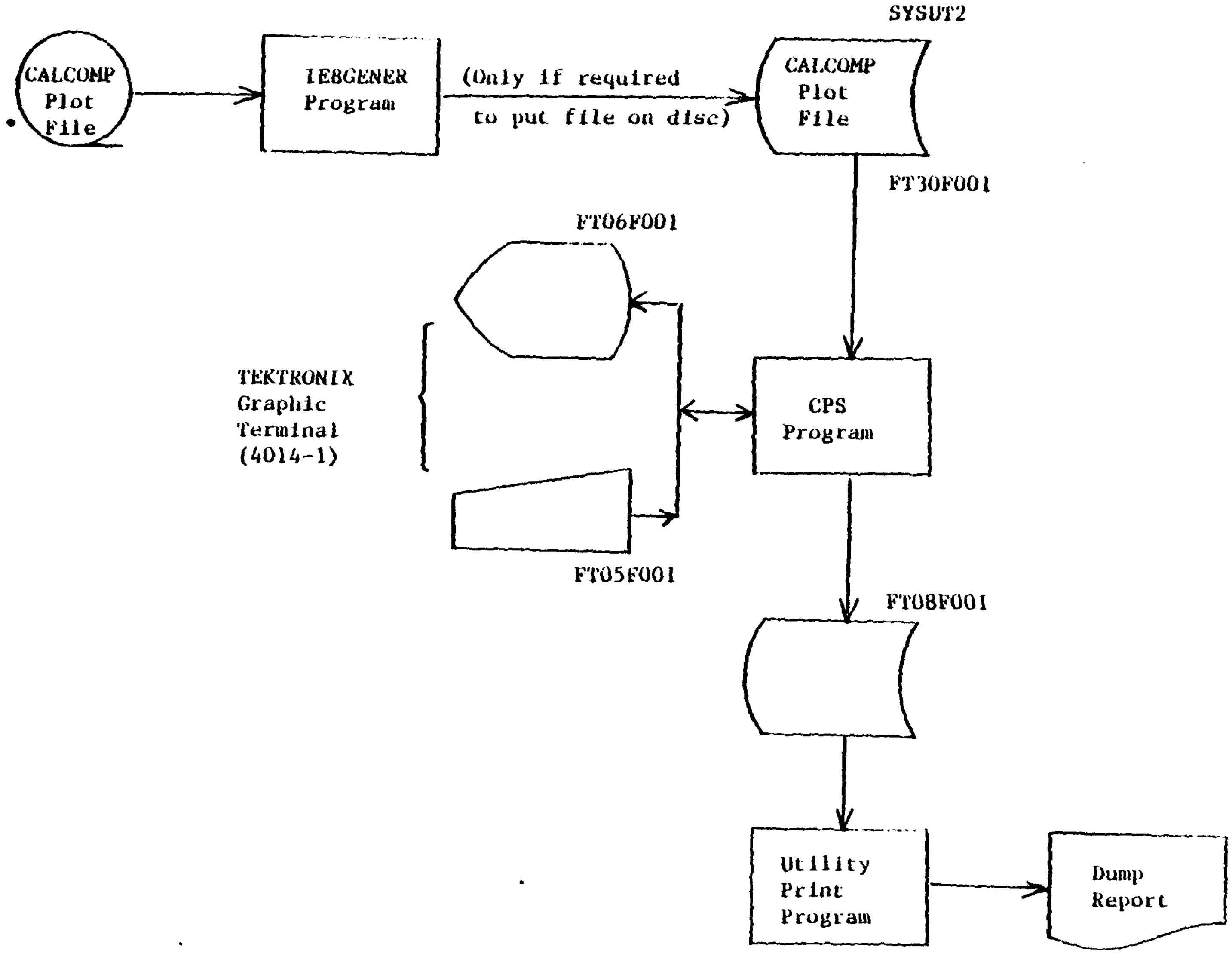


FIGURE 2-01: System Flowchart (NASC AS/5-3)

CALCOMP Plot Files: Filename FILEnn.DAT, where NN is user supplied
from the terminal.

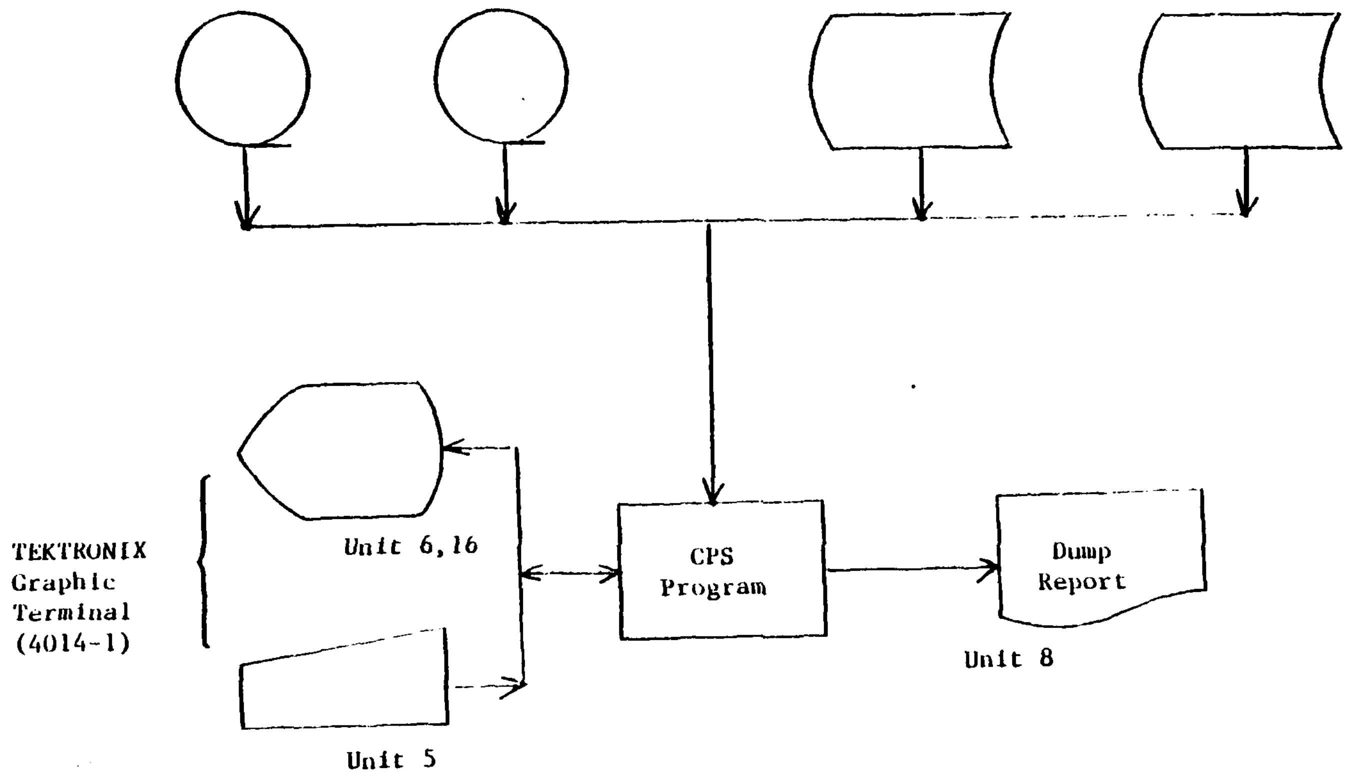


FIGURE 2-02: System Flowchart (PDP-11/70)

2.4 Program Description. The CPS includes one main program and 36 subroutines on the NASC AS/5. On the PDP-11/70, the CPS includes one additional subroutine (37 total).

2.4.1 Program MAIN.

a. Identification: MAIN.

b. Functions: MAIN performs major functions:
Initialization.
Accepting and interpreting user commands.
Setting global variables to execute user commands.
Plotting an input CALCOMP file, when commanded.
Dumping an input CALCOMP file when commanded.
Termination.

The program is written in FORTRAN IV, in a top-down fashion, so that most of the general and detailed functions are performed by subroutines, which may be grouped into four classes:

. Specialized subroutines written to support this project. These are, in effect, an extension of MAIN, and have been separated for ease of maintenance and understanding. See para 2.4.2.

. Service subroutines for mathematical computation. Most of these have been written to support MAIN and its specialized subroutines, but are called from various places in the program when needed. See para 2.4.3.

. String handling routines for character manipulation. See para 2.4.4.

. Graphics subroutines provided by the Tektronix Corporation to establish an interface between application programs and the Tektronix graphic display terminal. See para 2.4.5.

Virtually all of the communication among the specialized subroutines is done through one labeled COMMON block (COMBLK), which contains the global variables of the program. (See para 2.4.1e for a description of the labeled COMMON block and the global variables.)

c. Input.

(1) User commands are entered through logical unit 5 (usually allocated to the terminal). They are free form, and consist of lines of 60 bytes or less. See the user's manual (ref. 1.2e) for a detailed description of all user commands.

(2) Graphic input is entered through the Tektronix graphic terminal via subroutine SCURSR. This data consists of three values: one byte of data, entered from the terminal, and two integers - the X and Y coordinates of the screen cursor (controlled by the user) when the RETURN key is pressed. The program uses this facility to control certain aspects of the plotting.

(3) CALCOMP plot files are the main data input. These files are produced by CALCOMP subroutines called by other application programs. Each record contains a 4 byte 'sync code' - the hexadecimal characters X'1F19191F' followed by - CALCOMP 'sentence' in a modified ASCII code.

On the HSF NASC AS/5-3, they are in the format of records produced by the FORTRAN unformatted WRITE statement and have the following characteristics:

RFCFM = VBS
LRECL = 364
BLKSIZE = 368

In other words, they consist of the 4 byte sync code followed by 360 bytes of data. On the PDP-11/70, the

CALCOMP records are of fixed length format and consist of the sync code followed by 356 characters of data. See Appendix D and reference 1.2b for a detailed description of the input records and the CALCOMP sentences.

d. Processing.

(1) The MAIN program is initiated as a task from the operating system, and has no parameters. It first initializes global variables as necessary, and then enters the main loop of the program, which repeats until the user enters the command 'END' (which terminates the run).

(2) The main loop of the program prompts the user for input by printing 'COMMAND?' at the terminal, and receiving an input line from logical unit 5. Upon receiving an input line, it matches the input to the program commands, and takes the appropriate action. If the input cannot be matched to a command, the program prints a warning message and repeats the prompt message. After the action is complete, the program again prompts with 'COMMAND?' and awaits a line from logical unit 5. (Unless, of course, the preceding command entered was 'END', in which case the program exits the loop and terminates the run.)

(3) Four commands cause immediate action. They are 'PLOT', 'DUMP', 'HELP' (or '?'), and 'END'. 'END' sets the switch named 'END', which causes the exit from the main loop. 'HELP' branches to the internal procedure HELPER, which further analyzes the input and calls the external subroutine HELP to provide the user on-line assistance in using the program. 'PLOT' and 'DUMP' both access a CALCOMP plot file. Following either of these two commands, the program begins to access the plot file through the subroutine

FETCH and returns one CALCOMP sentence for each call, until end-of-file is encountered, and the program exits the loop. The bits PLOT and DUMP, which are set when the commands are interpreted, determine whether the sentence thus obtained is interpreted (PLOT is on) or listed on logical unit 8 (DUMP is on.)

If PLOT is on, the program calls subroutine FETCH to get the next sentence, calls DECODE to extract the values, and, if appropriate, calls DRAW to follow the instructions of the sentences.

(4) The remaining thirteen commands control or modify the performance of the next 'PLOT' command entered by changing the values of the global variables that the various subroutines use in performing their functions.

'BOX' causes the program to call the external subroutine BOX to designate the boundaries for the next plot directly on the screen. See para 2.4.2c.

'ROTATE' accepts the angle of rotation by calling external subroutine ASK, and then modifying the variables PSI, COSPSI, and SINPSI in response.

'SCALE' scales the plot by accepting a scale factor through external subroutine ASK, and then modifying the matrix CLCRNR.

'FAST' sets the bit EXACT off.

'EXACT' sets the bit EXACT on.

'WINDOW' causes the program to call the external subroutine WINDOW to allow the user to specify the limits of the plot by entering coordinates in inches.

'NOWINDOW' loads values into the variables in labeled COMMON that cause the entire CALCOMP plot bed to be mapped onto the terminal screen, reserving the lower part of the screen for operator messages. It then calls the subroutine DEFALT to initialize the next plot to default conditions.

'FRAME' sets the bit FRAME on.

'NOFRAME' sets the bit FRAME off.

'REORIGIN' sets the bits HALT and RRIGIN on.

'HALT' sets the bit HALT on.

'NOHALT' sets the bits HALT and RRIGIN off.

'FULLSCREEN' loads values into the variables in the labeled COMMON block which cause the entire vertical extent of the terminal screen to be scaled to the short extent of the CALCOMP plot bed. It then calls the subroutine DEFALT to initialize the next plot to default conditions.

See Table 2-01 for a logic diagram of program MAIN.

(5) Conventions - Statement labels are numbered in ascending order. The CONTINUE statement at the beginning of the main loop is #30. The statements before each IF are labeled from 50-900 in increments of 50. The statements controlling the PLOT/DUMP loop are labeled from 50-1100. Statements in the 2000's refer to internal subroutine SCALE. See

Table 2-02 for a logic diagram of SCALE. Statements in the 3000's refer to internal subroutine ROTATE. See Table 2-03 for a logic diagram of ROTATE. Statements in the 4000's refer to internal subroutine HELPER. Internal subroutines are implemented using the assigned GOTO.

(6) The program requires a TSO region of 128 bytes to execute successfully.

e. COMMON Block COMBLK and global variables. (All variables in COMMON are declared and initialized in a BLOCK DATA subprogram.)

In the following descriptions, reference is made to parameters extracted from CALCOMP sentences. In general, the data transmitted in this form is identified by letters that precede numeric or string data. The general content of the parameters is as follows:

A,B	Descriptors for dashed lines; ignored by this program, which uses Tektronix subroutine DASHA.
D	Transmits pen number, or pen-up/pen-down information.
E,F	Horizontal and vertical length, and hence implicitly, orientation, of characters to be plotted in strings.
G	Designates action to be taken.
I,J	Coordinates of the center of a circle.
M	Halt code.
N	Sentence number.
P,Q,R,S	The horizontal and vertical scaling matrix applied to CALCOMP coordinates to effect scaling and rotation within the CALCOMP system.
T	The search address, used for operator intervention.
U,V	Horizontal and vertical offsets applied by the CALCOMP system.
X,Y	Horizontal and vertical coordinates.

Additional explanations are contained in Reference 1.2b. The structure of the sentences within the records is described in Appendix D.

Global Variables:

AVALUE	The A parameter set by default or extracted from a CALCOMP sentence.
BDCRNR (3,2)	The coordinates, in CALCOMP units, of the points which correspond to the lower left, upper left, and upper right corners of the screen, respectively. Mnemonic: Bed-Corners.
BVALUE	The B parameter set by default or extracted from a CALCOMP sentence.
CHCRNR (3,2)	The coordinates, in inches, of the lower left, upper left, and upper right corners of the plotted area. Mnemonic: Inch-Corners.
CHVRTC (3,2)	The values of BDCRNR expressed in inches. Mnemonic: Inch-Vertices.
CLCRNR (3,2)	The coordinates, in CALCOMP units, of the lower left, upper left, and upper right corners of the plotted area. Mnomonic: CAL-Corners.
CLTTK (3,2)	The transformation matrix to go from the CALCOMP coordinate system used in the plot file to the Tektronix unit coordinate system passed to the Tektronix software. Mnemonic: CAL-to-Tek.
COSPHI	The cosine of PHI.
COSPSI	The cosine of PSI.

COSPSX	The cosine of PSIX.
COTPHI	The cotangent of PHI.
CSCPHI	The cosecant of PHI.
CURVE	A logical variable set on if the program is 'in a spline interpolation' and set off when the program 'goes out of spline interpolation'.
DFLTON	A logical variable used to communicate the source of the screen window now in use. It is on if the limits are set within subroutine DEFALT, or off if the proportions were established by the WINDOW command. Mnemonic: Default-On.
ENDFIL	A logical variable set on when endfile is reached on the plot file. Mnemonic: End-File.
EVALUE	The E parameter extracted from a CALCOMP sentence.
EXACT	A logical variable set on if curves are to be plotted as closely as possible, or off if they are to be approximated by line segments.
FILOPN	A logical variable set on if the plot file has been opened; off otherwise. Mnemonic: File-Open.
FRAME	A logical variable set on to cause the execution of the actions of the 'FRAME' command.
FVALUE	The F parameter extracted from a CALCOMP sentence.

GVALUE	The G parameter most recently extracted from a CALCOMP sentence.
HALT	A logical variable set on to cause the program to activate the screen cursor when a temporary halt code (M1) is encountered.
HEIGHT	The height of characters to be drawn by operator message request or symbol plots. It is derived from EVALUE, FVALUE, and NVALUE, and is stored as an integer multiple of 8.
INFILE	The logical unit number of the file to be plotted. For the NASC AS/5 it is defaulted to 30. For the PDP-11/70 it is supplied by the user.
IVALUE	The I parameter extracted from a CALCOMP sentence.
JVALUE	The J parameter extracted from a CALCOMP sentence.
LENSNT	The length of the CALCOMP sentence being processed, in bytes.
LETTER (26)	An array containing the letters of the alphabet.
LVALUE	The length of the character string used in symbol plots or operator message displays.
MVALUE	The halt code extracted from a CALCOMP sentence or reset to zero by the plot routines.
NSNTNC	The N parameter most recently extracted from a CALCOMP sentence.

NVALUE	The factor used by CALCOMP routines applied to compute the height of plotted characters. NVALUE equals 1.00 for noncentered characters and equals 1.875 for centered characters. See reference 1.2b for further information.
OLDX	The previous value of XTEK. Used to reset the X coordinate of the origin.
OLDY	The previous value of YTEK. Used to reset the Y coordinate of the origin.
PHI	The angle, expressed in radians, whose tangent is TANPHI. This angle, and its trignometric functions, govern the proportions of the plotting area used on the CRT face.
PLOTNG	A logical variable set on when the program begins processing a plot file. It is used to control the exit from the plotting loop. Mnemonic: Plotting.
PNDOWN	A logical variable set on if the action is to draw; off if it is to move. Mnemonic: Pen-Down.
PREVX	The previous value of XTEK. Used when value of XTEK changes.
PREVY	The previous value of YTEK. Used when value of YTEK changes.
PSI	The angle, in degrees, by which the plot or window is to be rotated about the origin.

PSIX	The angle, in radians, by which the previous plot was rotated about the origin.
PVALUE	The P parameter extracted from a CALCOMP sentence or set by default.
QVALUE	The Q parameter extracted from a CALCOMP sentence or set by default.
RRGIND	A logical variable set on either when the user enters "REORIGIN" or when a CALCOMP sentence contains a reorigin instruction. Mnemonic: Reorigin-Indicator.
RRIGIN	A logical variable set on to allow the user to control the origin of coordinates with the screen cursor. Mnemonic: Reorigin.
RVALUE	The R parameter extracted from a CALCOMP sentence or set by default.
SCCRNR (3,2)	The coordinates, in Tektronix units, of the lower left, upper left, and upper right corners of the CRT area available for plotting. Mnemonic: Screen-Corners.
SECPHI	The secant of PHI.
SINPHI	The sine of PHI.
SINPSI	The sine of PSI.
SINPSX	The sine of PSIX.

SNTNCE (128)	An array which contains the CALCOMP sentence being processed.
SPCIL	The ASCII character associated with the most recently received CALCOMP sentence containing a single symbol request (G53 code).
SPLLEN	The number of bytes of data in SPLTXT.
SPLTXT (64)	An array containing the untranslated ASCII character string transmitted with the most recent symbol plot, special symbol plot, or operator message request (G52, G53, or G55 code).
STEP	A logical variable set on when the current Tektronix coordinates are computed.
SVALUE	The S parameter extracted from a CALCOMP sentence or set by default.
TANPHI	The height of the available plotting area divided by the width of the available plotting area; i.e., the tangent of PHI.
TKCRNR (3,2)	The coordinates, in Tektronix units, of the lower left, upper left, and upper right corners of the plotted area. Mnemonic: Tek-Corners.
TVALUE	The search address from a CALCOMP sentence.
UVALUE	The U parameter extracted from a CALCOMP sentence or set by default.
VVALUE	The V parameter extracted from a CALCOMP sentence or set by default.

XBFORE	The previous value of XCAL.
XCAL	The most recently received X value in CALCOMP units, from a CALCOMP sentence.
XMAX	The largest rescaled* X value, in CALCOMP units, received during the current plot.
XMIN	The smallest rescaled* X value, in CALCOMP units, received during the current plot.
XOFST	Amount of the most recent displacement in the X direction, in CALCOMP units, due to reorigin. Mnemonic: X-Offset.
XORG	X-coordinate of the most recent origin before reorigin.
XTEK	The X coordinate, in Tektronix units, derived from the current values of XCAL, YCAL, and CLTTK.
YBFORE	The previous value of YCAL.
YCAL	The most recently received Y value, in CALCOMP units, from a CALCOMP sentence.
YMAX	The largest rescaled* Y value, in CALCOMP units, received during the current plot.
YMIN	The smallest rescaled* Y value, in CALCOMP units, received during the current plot.

YOFST	Amount of the most recent displacement in the Y direction, in CALCOMP units, due to reorigin. Mnemonic: Y-Offset.
YORG	Y-coordinate of the most recent origin before reorigin.
YTEK	The Y coordinate, in Tektronix units, derived from the current values of XCAL, YCAL, and CLTTK.

*Note: Rescaled, in the descriptions of XMAX, XMIN, YMAX, YMIN, means the X and Y values resulting after the current values of UVALUE, VVALUE, PVALUE, QVALUE, RVALUE and SVALUE are applied to the XCAL and YCAL values extracted from the CALCOMP sentences.

f. Local Variables:

(1) SCLRAT - Scale ratio that user enters as part of the "SCALE" command.

(2) CALCOMP Proportions - These variables are the values used when the CALCOMP plotter bed is mapped onto the terminal screen, reserving the lower portion of the screen for operator messages.

(a) CLSCCR (3,2) - The coordinates, in Tektronix units, of the lower left, upper left, and upper right corners of the CRT available for plotting. Mnemonic: CALCOMP-Screen-Corners.

Actual values: 0, 721
0, 3120
4095, 3120

(b) CLBDCR (3,2) - The coordinates, in CALCOMP units of the points which correspond to the lower left, upper left, and upper right corners of the screen respectively. Mnemonic: CALCOMP-Bed-Corners.

Actual values: 0, 0
0, 243840
416560, 243840

(c) CLCHVR (3,2) - The values of CLBDCR expressed in inches. Mnemonic: CALCOMP-Inch-Vertices.

Actual values: 0, 0
0, 48
82, 48

(d) CLPSI - The angle between the lower boundary of the plot area and the diagonal of the plot area expressed in radians. Mnemonic: CALCOMP-PSI.

Actual value: 0.52994

(e) CLTANP, CLCOTP, CLSINP, CLCOSP, CLSECP, CLCSCP are the tangent, cotangent, sine, cosine, secant, and cosecant respectively of PSI when CALCOMP proportions are in force.

Actual value:

Tangent: .5858364	Cosine: 0.8628374
Cotangent: 1.706961	Secant: 1.158697
Sine: 0.5054816	Cosecant: 1.978312

(3) Tektronix Proportions - These variables are the variables used when the plot is to use the entire screen of the Tektronix terminal.

(a) TKSCCR (3,2) - The coordinates, in Tektronix units of the lower left, upper left, and upper right corners of the CRT available for plotting. Mnemonic: Tektronix-Screen-Corners.

Actual values: 0, 0
0, 3120
4095, 3120

(b) TKBDCR (3,2) - The coordinates, in CALCOMP units, of the points which correspond to the lower left, upper left, and upper right corners of the screen respectively. Mnemonic: Tektronix-Bed-Corners.

Actual values: 0, 0
0, 243840
320040, 243840

(c) TKCHVR (3,2) - The values of TKBDCR expressed in inches. Mnemonic: Tektronix-Inch-Vertices.

Actual values: 0, 0
0, 48
63, 48

(d) TKPSI - The angle between the lower boundary of the plot area and the diagonal of the area, in radians. Mnemonic: Tektronix-PSI.

Actual value: 0.651077

(e) TKTANP, TKCOTP, TKSINP, TKCOSP, TKSECP, TKCSCP are the tangent, cotangent, sine, cosine, secant, and cosecant respectively of PSI when Tektronix proportions are in force.

Actual values:

Tangent:	.7619048	Cosine:	0.7954317
Cotangent:	1.3125	Secant:	1.257179
Sine:	.6060432	Cosecant:	1.6500473

(4) Keyword Variables - These variables are each set to the commands used by the CPS.

(a) KWPLOT - the word "PLOT"

(b) KWDUMP - the word "DUMP"

(c) KWEND - the word "END"

(d) KWQUES - the character "?"

(e) KWBOX - the word "BOX"

(f) KWROT - the word "ROTATE"

(g) KWSCAL - the word "SCALE"

(h) KWFAST - the word "FAST"

(i) KWEXAC - the word "EXACT"

(j) KWWIND - the word "WINDOW"

(k) KWNWIN - the word "NOWINDOW"

(l) KWHELP - the word "HELP"

(m) KWFRAM - the word "FRAME"

(n) KWNFRM - the word "NOFRAME"

(o) KWRORG - the word "REORIGIN"

(p) KWHALT - the word "HALT"

(q) KWNHLT - the word "NOHALT"

(r) KWFULL - the word "FULLSCREEN"

(5) REPLY2, REPLY4, REPLY5, REPLY6, REPLY7, REPLY8, REPLY9, and REPL11 are used to extract 2 character, 4 character, 5 character, 6 character, 7 character, 8 character, 9 character, and 11 character terminal responses respectively.

(6) LOCFLG - A logical variable which is turned off until user enters a correct value for SCALE or ROTATE, in which case it is turned on. Mnemonic: Local-Flag.

(7) BLKREP, BLKRP1, BLKRP2 - These variables are turned on when the user enters a blank value for SCALE or ROTATE. BLKRP1 and BLKRP2 are used because the PDP-11 Fortran compiler allows only 5 continuation cards, unless overridden, and more than 5 lines are needed to test the values of the REPLY array. Thus, BLKRP1 is turned on when the first 8 members of REPLY are blank; BLKRP2 is turned on when members 9-15 of REPLY are blank; BLKREP is turned on when both BLKRP1 and BLKRP2 are turned on.

2.4.2 Specialized Subroutines.

a. ASK

(1) Identification: ASK

(2) This subroutine is used for prompting within the program where needed. Given an integer parameter as argument, it prints the corresponding prompting message on the terminal. When the user enters a reply, the routine translates all lower case letters to upper case and returns the translated reply to the calling routine.

(3) Values of Integer Parameter:

"1" - Prompt saying "COMMAND?"
"2" - Prompt saying "ANGLE (DEGREES)?"
"3" - Prompt saying "SCALE FACTOR?"
"4" - Prompt saying "X1=?"
"5" - Prompt saying "Y1=?"
"6" - Prompt saying "X2=?"
"7" - Prompt saying "Y2=?"

b. BLANKR

(1) Identification: BLANKR

(2) Functions:

. Compute the transition matrix from CALCOMP coordinates to Tektronix by calling LLSQ, using the current values of CLCRNR and TKCRNR.

. Initialize the plot by blanking the screen and establishing window values through Tektronix subroutines.

. Draw the frame around the plot, if required.

. Replace the existing window parameters, by calling LLSQ and TRANS using the current values of CHCRNR, if required.

. Check for errors in the transformation specified, and terminate the plot if any are discovered.

(3) Processing: See Table 2-04 for a logic diagram of this subroutine.

c. BOX

(1) Identification: BOX

(2) Functions:

. Accept boundary data for the next plot directly from an existing plot, using the screen cursor.

. Display the boundaries computed from this data on the current plot, for user inspection.

(3) Processing:

. The routine assumes that the four courners of the CRT area being used are numbered 1 through 4, clockwise, starting at the lower left corner. It accepts screen coordinates corresponding to any two of them, in arbitrary order, and computes the other two. The first three thus computed are used to compute the corresponding values of CLCRNR, and all four are used to draw the boundaries of the next plot on the screen.

. The first part of the program uses the subroutine SCURSR to accept two character value and two pairs of coordinates from the screen. The character values must be distinct character integers, 1 through 4. The coordinates specified must be distinct. Any reply that does not meet these conditions is ignored, and a prompt 'INVALID REPLY' is printed. The user manipulates the screen cursor thumbwheels to place the crosshair at a desired point, and enters the numeral designating the corner desired. When two valid values have been accepted, the processing continues.

. The routine determines which two corners have been specified and computes the coordinates, in Tektronix units, of the two missing corners of the rectangle bounding the desired plot area. This computation uses the trigonometric functions specified in the labeled COMMON block, which then computes a rectangle whose aspect ratio is that chosen previously to be either that of the full CALCOMP bed (default or 'NOWINDOW') or that of the full Tektronix screen ('FULLSCREEN').

. When all four corners are known, the routine inverts the transformation for the first three points to establish new values for CLCRNR, and draws the rectangle boundary on the terminal screen. It then displays the screen cursor once more, to dump the Tektronix routines' buffer and to allow the user to control the location of the next printed output. See Table 2-05 for a logic diagram of BOX.

(4) Local Variables.

(a) POINTS(2) - The numbers of the corners that the user specified to enclose the box.

(b) X(4) - The X coordinates of the four corners of the box.

(c) Y(4) - The Y coordinates of the four corners of the box.

(d) RHO2, RHO3, RHO4, THETA2, THETA3, THETA4 - These variables are used to derive the coordinates of all four corners of the box when the coordinates of two corners are known.

(e) PNT12 - turned on if corners 1 and 2 were entered.

(f) PNT13 - turned on if corners 1 and 3 were entered.

(g) PNT14 - turned on if corners 1 and 4 were entered.

(h) PNT23 - turned on if corners 2 and 3 were entered.

(i) PNT24 - turned on if corners 2 and 4 were entered.

(j) PNT34 - turned on if corners 3 and 4 were entered.

d. CALTEK

(1) Identification: CALTEK

(2) Functions:

- Convert the CALCOMP coordinates given to the Tektronix coordinates needed for plotting.
- Establish values for the largest and smallest CALCOMP coordinates encountered during a plot.
- Establish the values of XBFORE and YBFORE, when a circle or circular arc is to be plotted, i.e. when GVALUE is 2 or 3.

(3) Processing:

- If GVALUE is 2 or 3, the current value XTEK and YTEK become the values of XBFORE and YBFORE.
- The rotations and offsets specified by PVALUE, QVALUE, RVALUE, SVALUE, UVALUE, and VVALUE are used to form intermediate variables XPRIME and YPRIME (the rescaled coordinates). The rescaled coordinates are still in CALCOMP units, and are the effective CALCOMP coordinates that were specified in the plot file.
- The rescaled coordinates are then mapped into Tektronix coordinates using the matrix CLTTK. Meanwhile, the bit STEP is turned on to indicate that the current Tektronix coordinates have been computed.
- The maximum and minimum values of the rescaled coordinates are updated.
- See Table 2-06 for a logic diagram of CALTEK.

e. CIRCLE

(1) Identification: CIRCLE

(2) Function: Draw circles or ellipses.

(3) Processing:

- Locate the center of the circle (XCENTR, YCENTR) using XBFORE and YBFORE, IVALUE, and JVALUE by adding XBFORE to IVALUE and YBFORE to IVALUE. This yields the coordinates of the center of the circle.
- Determine whether to draw a full or partial circle by comparing the distance from the center to (XBFORE, YBFORE) to the distance from the center to (XCAL, YCAL). If the distances are equal, draw a partial circle - if not, draw a full circle.
- Compute the value of the angular increment: If the logical variable EXACT is on, DELTA is the smallest increment

which will cause a Tektronix coordinate change of 1 unit. If not, it is 1/32d of the angle subtended by the circle.

The sign of DELTA is determined by the direction of the circle. If GVALUE is 2, the circle is clockwise, and DELTA is negative: If GVALUE is 3, the circle is counterclockwise and DELTA is positive.

- Compute coordinates in raw CALCOMP coordinates for points on the circle, call CALTEK to convert them to Tektronix coordinates, and draw to the points thus determined, until the circle is completed.
- See Table 2-07 for a logic diagram of CIRCLE.

(4) Local Variables:

(a) XCENTR - X coordinate for center of circle (integer).

(b) YCENTR - Y coordinate for center of circle (integer).

(c) DET - Determinant of matrix CLTTK (CAL-to-Tek transformation matrix).

(d) STANG - Starting angle.

(e) ENDANG - Ending angle.

(f) DELTA - Angular increment.

(g) GTANG - Logical variable turned on either when the ~~angel~~ angle is greater than the ending angle (when the increment is positive) or when the angle is less than the ending angle (when the increment is negative). Meaning: Greater angle.

(h) SIGN - The sign of the angular increment. Equals 1 when DELTA is positive. Equals -1 when DELTA is negative.

(i) CENTRX - X coordinate for center of circle (floating point).

(j) CENTRY - Y coordinate for center of circle (floating point).

f. DECODE

(1) Identification: DECODE

(2) Function: Extract values from CALCOMP sentences and place them into the appropriate global variables.

(3) Processing:

. One complete sentence is processed at a time. The routine scans the sentence, locating the letters that identify numeric parameter values. For each identifying letter, in turn, the numeric characters immediately following it are extracted and converted into numbers.

Depending upon the identifying letter, the appropriate global variable is assigned a new value. Generally, the global variable name is constructed from the identifying letter followed by 'VALUE', so that the value for J goes in JVALUE, etc.

. There are some exceptions:
D1 and D2 CALCOMP codes are used to set the bit PNDOWN on or off, respectively. When a G53 code (special character plot) is encountered, the value of SPCIL (special character) is used to set NVALUE to either 1.875 or 1.000 as appropriate.

When a G5 code is encountered, the GVALUE becomes 5 on the odd-numbered encounters, and 0 on even-numbered encounters. This is done because G5 codes come in pairs. The first of a pair begins a curve plot - the other ends it. See the discussion of subroutines DRAW and SPLINE for details.

. See Table 2-08 for a logic diagram of DECODE.

(4) Local Variables:

(a) INTVAL - current integer value within CALCOMP system.

(b) Pointer variables - below is list and meanings:

PNTR - Pointer to current letter code

NEXT - Pointer to next letter code

PNTRHD - Hold area for PNTR

LEN - Length of sentence between PNTR and end of sentence.

(c) ENDFND - logical variable that is turned on when end of CALCOMP sentence is found.

(d) RSTSEN(128) - Remainder of CALCOMP sentence from PNTR to the end. Mnemonic: Rest-of-Sentence.

g. DEFALT

(1) Identification: DEFALT

(2) Function: Establish the default scaling and rotation parameters.

(3) Processing: Default values are assigned to the scaling and rotation parameters. The bit DFLTON is set on to indicate default scaling and rotation parameters are being used. A call to OPMSG is made to initialize the operator message parameters.

(4) See Table 2-09 for a logic diagram of DEFALT.

h. DRAW

(1) Identification: DRAW

(2) Functions:

. Interpret the command words contained in the data extracted from the current CALCOMP sentence, and call the subroutine required to take the action.

. At the end of each plot, set up the parameters needed for optimal plotting of the same file.

(3) Processing:

- Obtain new variable values by calling DECODE and CALTEK.
- Inspect MVALUE for non-zero halt codes.
- If MVALUE is 1, a temporary halt has been commanded. If the bit HALT is on, activate the screen cursor to accept one byte of data, plus coordinates, from the terminal. If the byte is a p or a P, suspend the plot by turning the bit PLOTNG off. If the byte is a period (.) stop the plot by calling WRAPUP, an internal procedure. Reset MVALUE to zero.
- If MVALUE is 2, reset MVALUE to zero, then stop the plot by calling WRAPUP.
- If MVALUE is zero, inspect GVALUE for the action to be taken, and take them. The values of GVALUE and the actions taken are as follows:
  - 1 Draw or move to the current coordinates (XTEK, YTEK).

    2,3 Call CIRCLE to draw circles and circular arcs.

    5 Call SPLINE to plot curve segments. Since the coordinates used in a spline plot are preceded and followed by a G5 sentence, the function of the G5 code is to start the spline interpolation, or to stop it, alternately. This program accomplishes this by code both in DECODE and here. Within DECODE, if a G5 code is encountered, and GVALUE is not 5, it is set to 5. If GVALUE is 5 it is set to zero. Therefore GVALUE is 5 only when spline interpolation is going on. The call to SPLINE sets the bit CURVE on to indicate spline interpolation (used later).

    52 Call STNDRD to plot a character sting.

    53 Call SPECL to plot a single special character.

. 25 Effect a reorigin of the coordinate system, by replacing the last row in the array CLTTK. If the bit RRGIND (reorigin indicator) is not yet turned on, turn it on, set the values of XOFST and YOFST to XBFORE and YBFORE, and set XORG and YORG to the last row of the array CLTTK (current origin). If the bit RRIGIN for a user desired reorigin is on, the new values for the origin are the X, Y coordinates returned from the screen cursor. If not, they are the present values of OLDX, OLDY.

. The external subroutine WRAPUP is used to terminate a plot and set optimal values for replotting the same file. The maximum and minimum values of XTEK and YTEK have been collected in XMAX, YMAX, XMIN and YMIN. They are compared to the proportions of the screen that are currently in use, and appropriate values are assigned to the array CLCRNR. WRAPUP is called as a separate subroutine.

. See Tables 2-10 and 2-11 for logic diagrams of DRAW and WRAPUP respectively.

i. FETCH

(1) Identification: FETCH

(2) Function:

Handles all communication with the plot files; includes opening and closing the file, reading records, and extracting CALCOMP sentences from them. The functions are controlled by an integer argument. When it is zero, the procedure places the next CALCOMP sentence into the global variable SNTNCE, in character form. If a quoted string is encountered, its CALCOMP ASCII value is placed into SPLTXT.

In addition, if the string is only one byte long, its CALCOMP ASCII value is placed into the lower-order byte of SPCIL.

(3) Processing:

FETCH has one argument, IFUNCT, which has three possible values. When IFUNCT = 0, the next CALCOMP sentence is retrieved. When IFUNCT = 1, the plot file is opened. When IFUNCT = 2, the plot file is closed. If none of the above values are present, the program will stop with a code of 1040. The formation of a CALCOMP sentence from the data in the plot file records depends upon the following observations about the structure of the records and the structure of the sentences:

Each record is a fixed length with the first four bytes containing '?99?', the 'sync code', with the remaining bytes of the record containing the data.

The end of the data is marked by the character '$' not in a quoted string.

A quoted string is delimited by the character '!' in pairs (for example, the string - !THIS IS A HEADING!).

The character '!' may appear in a quoted string only if the string is one byte long — resulting in the pattern '!!!' transmitting the character '!'.

A CALCOMP sentence is terminated by a period ('.') outside of a quoted string (for example, the sentence N25D2XY. terminates with a period).

CALCOMP sentences may be broken up over more than one physical record.

See Appendix D for further description of a CALCOMP file and examples.

The routine maintains a set of bit variables and integer pointers as place and status markers to allow for various configurations of sentences in records.

The variables are:

RCDNDD - Requests that a record be read. Mnemonic: Record-Needed.

SCANST - Offset of the next uninspected byte. Mnemonic: Scan-Start.

BLDST - Offset of the next byte to be placed into the sentence. Mnemonic: Build-Start.

INSTRG - A quoted string delimiter has been found on the left.

TERMNT - Offset of the next '.' after SCANST.

BREAK - Offset of the next '$'.

STRSTT - Offset of the next '!'.

EXCL - Equal to '!'. On the NASC, exclamation points cannot be printed.

The processing proceeds by using the aforementioned variables and the substring functions to control the formation of the sentence. The code is straightforward. Note that the sentence is restricted to be 512 bytes long, and the quoted strings are restricted to 255 bytes. This is expected to be adequate for any realistic use of the plotter. See Table 2-12 for a logic flow of FETCH.

j. HELP

(1) Identification: HELP

(2) Function: Place program messages on logical unit 6.

(3) Processing:

HELP has one argument, MSGNO, which determines the message to be printed. Actual message numbers are listed below:

MSGNO=1 - Prints out opening message.

MSGNO=2 - Prints out list of available commands.

MSGNO=3 - Prints out information on how to use "PLOT" command.

MSGNO=4 - Prints out information on how to use "DUMP" command.

MSGNO=5 - Prints out information on how to use "SCALE" command.

MSGNO=6 - Prints out information on how to use "ROTATE" command.

MSGNO=7 - Prints out information on how to use "WINDOW" command.

MSGNO=8 - Prints out information on how to use "NOWINDOW" command.

MSGNO=9 - Prints out information on how to use "HALT" command.

MSGNO=10 - Prints out information on how to use "NOHALT" command.

MSGNO=11 - Prints out information on how to use "END" comand.

MSGNO=12 - Prints out information on how to use "NOFRAME" command.

MSGNO=13 - Prints out information on how to use "FRAME" command.

MSGNO=14 - Prints out information on how to use "BOX" command.

MSGNO=15 - Prints out information on how to use "REORIGIN" command.

MSGNO=16 - Prints out information on how to use "FAST" command.

MSGNO=17 - Prints out message saying "INVALID COMMAND".

MSGNO=18 - Prints out information on how to use "EXACT" command.

MSGNO=19 - Prints out information on how to use "FULLSCREEN" command.

k. OPMSG

(1) Identification: OPMSG

(2) Functions: Process operator messages, or reset initial conditions for them.

(3) Processing:

OPMSG has one argument, FUNCT. When FUNCT = 0, the initial conditions for the messages are reset; otherwise operator messages are processed. The routine sets the character size based upon the length of the operator messsge. It computes the location for each line to be printed, starting below the default bottom limit at the line Y=720 on the CRT screen, writes the message at the appropriate location, and activates the screen cursor.

If 'RRIGIN' is on, it resets the origin of coordinates by replacing the last row of CLTTK with the values of IX and IY returned from the screen cursor. It then resets the screen window to its former values and returns. See Table 2-13 for logic flow of OPMSG.

(4) Local Variables:

(a) XSTR(65) - Stores contents of the operator message to be displayed on the terminal.

(b) LINE - The actual line on the terminal where the message is to be displayed, normally at the bottom of the terminal.

(c) LINES - The number of lines in the operator message.

(d) POINTS - Number of points already plotted.

1. SPECL

(1) Identification: SPECL

(2) Function: Plot a single special character on the screen.

(3) Processing:

- Call SYMHGT to establish the orientation of the character in the variables CSTHET and SNTHET.

- If invalid values are detected for CSTHET, SNTHET, or HEIGHT, the program returns to the caller.

- Otherwise, it draws the character, using the value of SPCIL as an index to the array POINT, which, in turn, contains pointers to the array ENTRY, which gives drawing directions for each character.

- For any given special character, (SPCIL), POINT (SPCIL + 1) is the subscript of the first value in ENTRY for that character. POINT (SPCIL + 2) is the subscript of the value in ENTRY following the last value for that character. For ease of addressing, PSTART is set to POINT (SPCIL + 1) and PBND is set to POINT (SPCIL + 2) - 1. Note, the subtraction of 1 is needed in order to get the last value for a given character since POINT (SPCIL + 2) points to the first value in ENTRY for the next character.

- The values in ENTRY are in pairs. They give the direction and length of the next move or draw required by the character.

  The X value is first, followed by the Y value. If the X value is 100 or more in absolute value, it is regarded as a move -- otherwise it is a draw.

- The length of the moves and draws are integers from 0 to 8, inclusive. A move is coded, in the X value, with an absolute value 100 greater than its true value, and with the appropriate sign.

- The signs are: + for up or right.
  - for down or left.

- The characters are drawn on an 8 by 8 unit grid, which is magnified by HEIGHT and rotated using CSTHET and SNTHET to compute new values for XCAL and YCAL.

- CALTEK is used to convert the XCAL and YCAL values to XTEK and YTEK.

- The draw or move, as appropriate, is done.

- For example, the letter 'A' has the ASCII value 49 when plotted as a special character. The corresponding values are:

  POINT (50) = 767

  POINT (51) = 783

  ENTRY (767) to ENTRY (782) are

  0,6,2,2,2,0,2,-2,0,-6,-106,4,6,0,-106,-4

  See Figure 2-03 to see how the plot works.

- The first 16 values are centered, and the remaining 48 values are started from the lower left corner. In either case, the beam returns to the starting position after the character is complete.

- See Table 2-14 for logic diagram of SPECL.

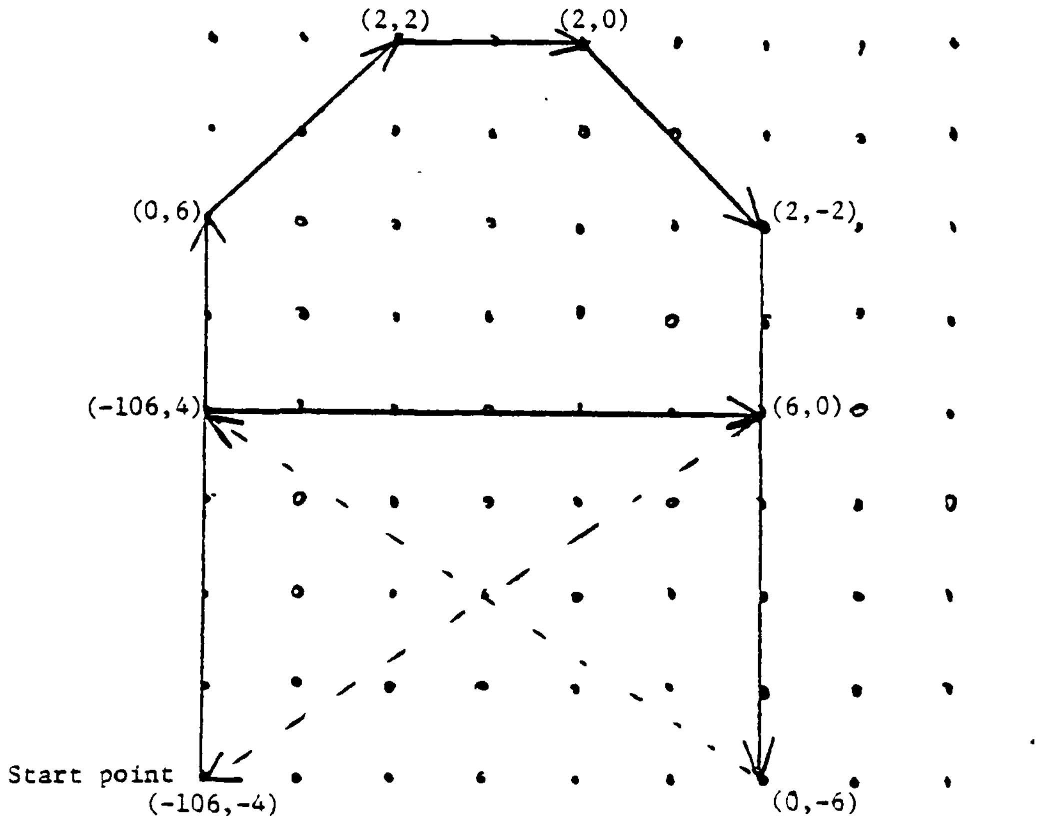
(4) Local Variables:

Due to limitations in the number of continuation statements allowed in FORTRAN, several smaller arrays have been set equal to the actual values of the arrays POINT and ENTRY. The values are placed in POINT and ENTRY through equivalence statements.

(a) POINT(65) - Contains pointers to the array ENTRY. The small arrays used to initialize POINT are described below:

PNTSPC(32) - Contains pointers to the array ENTRY for all characters whose ASCII representation is less than the ASCII representation for zero. Mnemonic: Pointer-to-Special-Characters.

PNTNML(33) - Contains pointers to the array ENTRY for all characters whose ASCII representation is greater than or equal to that of zero. Includes pointers to all numerics and alphabetic characters. Mnemonic: Pointer-to-Normal-Characters.



The values of ENTRY for the letter A among the special characters are:
(0,6,2,2,2,0,2,-2,0,-6,-106,4,6,0,-106,-4)
resulting in the moves and draws shown above.

DRAW ————————————

MOVE — — — — — — — — — — — —

Note: Same technique is used to draw standard ASCII characters and special characters.

Figure 2-03: Drawing the Letter 'A'

(b) ENTRY(973) - Gives drawing directions for each special character. The small arrays used to initialize ENTRY are described below:

LSBLK1(66), LSBLK2(58), LSBLK3(76) - Contains drawing directions for special characters whose ASCII representation is less than the representation for a blank. Mnemonic: Less-than-blank.

BLKCHR(6) - Contains drawing directions for blank characters. Mnemonic: Blank Character.

SPCHR(58), SPCHR2(64), SPCHR3(74), SPCHR4(24) - Contains drawing directions for all special characters whose ASCII representation is greater than that of blank and less than that of zero. Mnemonic: Special-Characters.

NUM03(80) - Contains drawing directions for the numeric characters from 0 to 3.

NUM46(58) - Contains drawing directions for the numeric characters from 4 to 6.

NUM79(64) - Contains drawing directions for the numeric characters from 7 to 9.

NUMSYM(72) - Contains drawing directions for all special characters whose ASCII representation is greater than the one for 9 and less than the representation of the "at sign",@. This array is called NUMSYM because it contains drawing directions for numeric symbols used in comparisons.

QUESAT(66) - Contains drawing directions for the question mark and the at sign.

LETAE(82) - Contains drawing directions for the letters from A through E.

LETFJ(70) - Contains drawing directions for the letters from F through J.

LETKO(60) - Contains drawing directions for the letters from K through O.

(c) SNTHET, CSTHET - The sine and cosine of the angle of orientation for the special character to be printed. These arguments are returned from subroutine SYMHGT.

(d) DELTAX, DELTAY - Change in the X and Y directions respectively, while character is being drawn.

(e) PSTART - The subscript of the first value in ENTRY for a given character. Equal to POINT (SPCIL +1). Mnemonic: Pointer-Start.

(f) PEND - The subscript of the last value in ENTRY for a given character. Equal to POINT (SPCIL +2) -1. Mnemonic: Pointer-Bound.

m. SPLINE

(1) Identification: SPLINE

(2) Function: Simulate the spline interpolation function of the CALCOMP plotter.

(3) . The logic of this subroutine is governed by the sequence of sentences produced for this function in the CALCOMP file. Spline interpolation is initiated by a G5 code, then the necessary coordinate pairs are supplied in following sentences, and finally a second G5 code terminates the interpolation. This is implemented in subroutine DECODE by setting GVALUE to 5 on the first G5 call and to zero on the second; i.e., to 5 on the odd-numbered occurrences

of the G5 code and to zero on the even-numbered occurrences thereafter.

The effect of this is to turn spline interpolation "on" and "off" by alternate occurrences of the G5 code in sentences.

Once the G5 code is detected, SPLINE is called for the sentence containing the G5 code and each subsequent sentence until GVALUE is set back to zero. The first call to SPLINE on a new interpolation is accompanied by the previous values of XTEK and YTEK, so that five calls are needed before the first interpolation takes place. After that, each subsequent call results in another segment of the curve being plotted.

. If the parameter FUNCT is zero, reset all variables to initial conditions, and return. (This is turning the spline function off.)

. The following is what happens when FUNCT is non-zero.

. Increment COUNT by 1, and store the current values of XTEK and YTEK in a work vector.

. If COUNT is 5, then the plotting is effected. If the flag EXACT is .TRUE., then the interpolated values are computed using a polynomial fit with a step size just large enough to provide a movement of one TEKTRONIX unit at each step. If EXACT is .FALSE., then the interpolation is omitted, and a straight line is drawn between the points.

. In either case, the latest four points are moved up, so that the oldest point is discarded. Then the interpolation is done from the second to the third point.

If the leading two points or the trailing two points are not distinct, then the curve interpolation is done by a quadratic fit. Otherwise, a cubic fit is used. The subroutine solves the interpolation problem by computing a vector of coefficients in terms of a parameter TX, which is then varied from -1 to +1 in appropriate steps to compute the intermediate points. Then COUNT is set to 4 so that it will become 5 on the next call to this routine, and the subroutine is terminated.

- See Table 2-05 for a logic diagram of SPLINE.

(4) Local Variables:

(a) RUNX(5) - Stores X coordinates of the current set of points to be plotted with interpolation being performed between the points whose X coordinates are represented by the second and third members of this array. The first and fourth members are used for curve fitting. At the beginning of each call to SPLINE, the desired X coordinates are stored in the second through fifth members, but they are all moved to the previous spot in the array before interpolation begins.

(b) RUNY(5) - Stores Y coordinates of the current set of points to be plotted with interpolation being performed between the points whose Y coordinates are represented by the second and third members of this array. The first and fourth members are used for curve fitting. At the beginning of each call to SPLINE, the desired Y coordinates are stored in the second through fifth members, but they are all moved to the previous spot in the array before interpolation begins.

(c) COUNT - Count of the number of points placed in RUNX and RUNY. Used to control moves and draws. When COUNT=3, the plotter beam is moved to the starting point of where the curve is to be plotted. When COUNT=5, the plot between any pair of points is being performed, after which COUNT is reset to 4, in preparation for the next interpolation.

(d) X1 - The X coordinate of the starting point of the curve segment where interpolation is to be done. Equal to RUNX(2).

(e) Y1 - The Y coordinate of the starting point of the curve segment where interpolation is to be done. Equal to RUNY(2).

(f) X2 - The X coordinate of the ending point of the curve segment where interpolation is to be done. Equal to RUNX(3).

(g) Y2 - The Y coordinate of the ending point of the curve segment where interpolation is to be done. Equal to RUNY(3).

(h) QUADCO(3,2) - Stores coefficients needed for quadratic interpolation. Each member QUADCO (*,1) stores coefficients for X while each member QUADCO (*,2) stores coefficients for Y.

(i) CUBCOF(4,2) - Stores coefficients needed for cubic interpolation. Each member CUBCOF (*,1) stores the coefficients for X while each member CUBCOF (*,2) stores the coefficients for Y.

(j) TX - A number between -1 and 1 used as a parameter in calculating the X and Y coordinates of any point where the small line segments that approximate a curve are drawn during interpolation.

(k) T2MAT(3,3) - Parameter matrix of coefficients used in conjunction with TX in quadratic interpolation.

(l) T3MAT(3,3) - Parameter matrix of coefficients used in conjunction with TX in quadratic interpolation.

(m) T4MAT(4,4) - Parameter matrix of coefficients used in conjunction with TX in cubic interpolation.

(n) XCOEFF and YCOEFF - Coefficients in for X and Y respectively, used in interpolating.

(o) XCORD and YCORD - The X and Y coordinates of the intermediate endpoint where a small line segment is to be drawn during the interpolation process.

n. STNDRD

(1) Identification: STNDRD

(2) Function: Plot a string of ASCII characters on the CRT.

(3) Processing:

. Call SYMHGT to establish the orientation of the characters in the variables CSTHET and SNTHET.

. If any invalid values are detected, the program returns to the caller without taking any other action.

- Otherwise, it draws each character in the string. The number of characters to be drawn is in COMMON in LVALUE. The technique for drawing the characters is identical to that used for subroutine SPECL (para 2.4.2 1 above) except that the beam is positioned at the lower right hand corner of the 8 by 8 grid after each character is drawn, and is therefore positioned properly to start the next character.
- See Table 2-16 for a logic diagram of STNDRD.

(4) Local Variables:

Due to limitations in the number of continuation statements allowed in FORTRAN, several smaller arrays have been set equal to the actual values of the arrays PNTR and ENTRY. The values are placed in PNTR and ENTRY through equivalence statements.

(a) PNTR(65) - Contains pointers to the array ENTRY. The small arrays used to initialize PNTR are described below:

PNTR1(36) - Contains pointers to the array ENTRY for the first 36 standard characters.

PNTR2(29) - Contains pointers to the array ENTRY for the last 29 standard characters.

(b) ENTRY(994) - Gives drawing directions for each standard character. The small arrays used to initialize ENTRY are described below:

LOWCH1(52), LOWCH2(56), LOWCH3(52), LOWCH4(60), LOWCHR(22) - Give drawing directions for all standard characters whose ASCII representation is less than the representation for zero.

NUM03(80) - Gives drawing directions for the numeric characters from 0 to 3.

NUM47(68) - Gives drawing directions for the numeric characters from 4 to 7.

NUM89(54) - Gives drawing directions for the numeric characters from 8 to 9.

NUMCHR(72) - Gives drawing directions for all characters whose ASCII representation is greater than the representation for 9 and less than representation for the at sign. This array is called NUMCHR because it continues many of the numeric comparison operators.

QUESAT(66) - Gives drawing directions for the question mark and the at sign.

LETAE(82) - Gives drawing directions for the alphabetic characters from A through E.

LETFJ(70) - Gives drawing directions for the alphabetic characters from F through J.

LETKO(60) - Gives drawing directions for the alphabetic characters from K through O.

LETPS(74) - Gives drawing directions for the alphabetic characters from P through S.

LETTW(52) - Gives drawing directions for the alphabetic characters from T through W.

LETXZ(36) - Gives drawing directions for the alphabetic characters from X through Z.

BEYLET(38) - Gives drawing directions for all characters whose representation is greater than the letter "Z". Mnemonic: Beyond-Letters.

(c) K1 - Equal to the internal representation of the character to be plotted. Equivalent to K1UP(2) with K1UP(1) equal to the high order portion of K1. Used to calculate PTR, the pointer used to address the PNTR array.

(d) PTR - The pointer to the array PNTR.

(e) PSTART, PBND, DELTAX, DELTAY, XSTHET, SNTHET are used in the same way as they are in subroutine SPECL which is described in para 2.4.2f.

o. WINDOW

(1) Identification: WINDOW

(2) Function: Allow the user to specify the boundaries of the plot in terms of inches. This simulates the WINDOW command of the CALCOMP plotter.

(3) Processing:

The program prompts the user for four values, which are the X,Y coordinates of the lower left and upper right corners of the rectangular boundaries of the plot. It then computes the corresponding coordinates in CALCOMP units and in Tektronix units, and places the resulting values into the arrays CLCRNR and TKCRNR in COMMON, where they govern the next plot. Finally, the signs of PSI and SINPSI are reversed, so that any existing rotation (entered before calling WINDOW) is applied to the area defined by WINDOW the same as it is on the CALCOMP plotter. See Table 2-17 for logic diagram of WINDOW.

(4) Local Variables:

(a) VALUE(4) - The four values (i.e. two points) of window data that the user entered. Below is a table of values and their equivalents:

VALUE(1) = X1

VALUE(2) = Y1

VALUE(3) = X2

VALUE(4) = Y2

(b) FLAG - A logical variable that is turned on each time the user enters a valid window value. Turned off before each prompt.

(c) BLKREP, BLKREP1, BLKREP2 - Logical variables that are turned on when user enters blanks as one of the four window values. Used the same way as the variables with the same name in program MAIN described in Section 2.4.1.

p. GETFIL

(1) Identification: GETFIL

(2) Function: Allows the user on the PDP-11/70 to specify the file he wishes to plot. Also, closes any previously opened plot file. Determines whether the file is accessed from disk or tape by setting INFILE.

(3) Processing:

GETFIL has one argument, IFUN. When IFUN equals 0, the routine closes any previously opened plot file and returns to FETCH. Otherwise, the program prompts the user for a logical unit number. If the file number is only one digit, the program inserts a leading zero. The program then builds a data set name of the form FILEnn.DAT, where nn is the logical unit number, and opens the file of that name. If any error is encountered, the program displays an error message, changes the value of INFILE to the one not yet used, thus testing the other device type, and tries the OPEN again. If the file cannot be opened on either tape or disk, the routine prompts for a new file number, until it opens a file successfully. INFILE is 30 for tape, and 25 for disk files.

2.4.3 Service Subroutines. There are three service subroutines used for mathematical computations. Two were written for this program, and one was selected from the IBM Scientific Subroutine Package.

a. LLSQ - This routine is from the IBM Scientific Subroutine Package. Its intended use is to solve the linear least squares problem for matrix arguments. In this program, it is used to derive the transition matrix CLTTK when those corresponding sets of coordinates, in CALCOMP units and TEKTRONIX units, have been established. See Appendix A for a mathematical description of the transformations. See Reference 1.2d for a complete description of LLSQ.

b. SYMHGT

(1) Identification: SYMHGT

(2) Functions: SYMHGT uses its input parameters (the current values of EVALUE, FVALUE and NVALUE) to compute three output parameters used in plotting characters or character strings. The output parameters are the symbol height, and two rotation parameters (cosine theta and sine theta). The symbol height is an integer, rounded up from the calculated height/8. This insures that the values returned will work with the two character plotting routines SPECL and STNDRD which draw characters on a grid of 8 x 8 points. See Figure 2-03.

c. TRANS

(1) Identification: TRANS

(2) Function: TRANS inverts the linear transformation used in SCOPE. That is, given a pair of points (U,V) in coordinate system B, and the 3 X 2 transition matrix which carries a point (X,Y) from coordinate system A

into coordinate system B, it replaces (U,V) with the preimage point (X,Y). See Appendix A for a description of the transformation used.

2.4.4 String Handling Routines. The following 8 subroutines are used to perform character string manipulation for the CALCOMP Preview System. On the NASC system, these routines are written in IBM assembler, while on the PDP-11, they are written in PDP Fortran 4+.

INDEX	VERIFY
SBSTRI	TRNSLT
SBSTRO	GSTRE
CONCAT	GSTRL

Below is a description of each routine, its calling sequence, and the abend code it returns when invalid parameters are encountered. On the NASC, the abend code is the completion code for an ABEND, while on the PDP-11, the result is a STOP N, with N being the abend code. INDEX, VERIFY, and GSTRE are functions which return an Integer *4 value. GSTRL is a function which returns a real value. The other routines are subroutines.

INDEX - An integer function which searches a string for a specified bit or character configuration. If the configuration is found, the starting position of the left most configuration within the string is returned. Otherwise, the value zero is returned.

Example calling sequence: I = INDEX (STR1, L1, STR2, L2)

STR1 is the string to be searched; L1 is the length of the portion of STR1 that is to be searched; STR2 is the character configuration desired, and L2 is the length of STR2 being dealt with in the search. The abend code is 1003.

SBSTRI - A subroutine which replaces a certain number of characters in one string with the same number of characters from another string.

Example calling sequence: CALL SBSTRI (STR1, IP1, L1, STR2)

STR1 is the receiving string, IP1 is the position in STR1 where character replacement is to begin, L1 is the number of characters to be replaced, and STR2 is the replacement string. No blank padding is done. The abend code is 1002.

SBSTRO - A subroutine which replaces the beginning characters of one string with the same number of characters from another string. If the number of characters to be replaced is not an even multiple of 4, the remaining bytes, up to an even multiple of 4, are replaced with blanks.

Example calling sequence: CALL SBSTRO (STR1, IP1, L1, STR2)

STR1 is the replacement string; IP1 is the starting position of STR1 being used in replacement; L1 is the number of characters being replaced, and STR2 is the receiving string. The abend code is 1001.

CONCAT - A subroutine which joins together two strings and places the results into a third string. It pads with blanks, if needed, up to a multiple of 4 bytes.

Example calling sequence: CALL CONCAT (STR1, L1, STR2, L2, STR3)

STR1 is the first string to be joined; L1 is the length of STR1; STR2 is the second string to be joined; L2 is the length of STR2; STR3 is the replacement string. The abend code is 1005.

VERIFY - An integer function which examines two strings to verify that each character in one string is represented in another string, returning a value of zero, if that is the case. Otherwise, the value returned is the position of the first byte in the first string that is not represented in the second string.

Example calling sequence: I = VERIFY (STR1, L1, STR2, L2)

In this case, VERIFY returns the first character in the first L1 bytes of STR1 that is not found in the first L2 bytes of STR2, or zero, if all bytes of STRL are also in STR2. The abend code is 1004.

TRNSLT - A subroutine which replaces a certain number of characters of a certain string with one of two designated tables. One table results in converting CALCOMP ASCII to the machine representation of the computer being used; while the other results in lower case letters to upper case.

Example calling sequence: CALL TRNSLT (STR1, L1, ITAB)

In this case, TRNSLT translates the first L1 bytes of STR1 with the designated table:

If ITAB = 1, translate CALCOMP ASCII to the proper computer representation. On the NASC, CALCOMP ASCII is converted to EBCDIC. On the PDP-11 CALCOMP ASCII is converted to PDP-11 ASCII.

If ITAB = 2, translate lower case letters to upper case.

The abend code is 1006.

GSTRE - A function which converts a numeric integer character string into an integer. This implements FORTRAN I format with length L1.

Example calling sequence: I = GSTRE (STR1, L1)

In this case, GSTRE returns the first L1 bytes of STR1 as an integer. The abend code is 1007.

GSTRL - A function which converts a numeric fixed character string into a floating point real value. This implements FORTRAN F format with length L1.

Example calling sequence: X = GSTRL (STR1, L1)

In this case, GSTRL returns the first L1 bytes of STR1 as a floating point value. The abend code is 1008.

2.4.5 Graphic Subroutines.

a. Identification: The following 10 subroutines from the Tektronix Corporation PLOT 10 control system are used in SCOPE.

ANMODE	AOUTST	DASHA	DRAWA	DWINDO
INITT	PNTABS	SCURSR	TERM	TWINDO

b. Functions: These subroutines provide the interface to the Tektronix graphic terminal. For details, see Reference 1.2c.

2.5 Program Logic Descriptions.

The CALCOMP Preview program was designed, and the first NASC version was written, using structured programming techniques available in the PL/1 language. The production program described in this manual is written in FORTRAN IV, conforming to ANSI standards where practicable. Those necessary functions that could not be performed using ANSI FORTRAN were implemented in IBM assembly language for the NASC system and in RSX-11M Fortran 4+ for the PDP-11.

The FORTRAN IV language does not contain the structured programming control logic available in block structured languages such as PL/1, PASCAL, FORTSIM, or FORTRAN 77. This frequently obscures the underlying program logic in the complexity of FORTRAN GOTO statements, computed GOTOs, etc.

The tables which follow contain logical descriptions of the more complex programs and subroutines in a pseudo-code that supplements the descriptions in para 2.4. The logic flow is shown by the control structures which are described below. Each description provides an example of how it is usually implemented in FORTRAN IV. A programmer who must change part of the program logic can relate the mnemonic names to the short FORTRAN variable names and the block structure to the FORTRAN implementation to locate the parts of the FORTRAN code that should be changed.

A. IF-THEN-ELSE

In a block structured language, this would be written as:

```
  IF (A) THEN
    (code for IF condition)
  ELSE
    (code for NOT IF)
  ENDIF
```

In FORTRAN, this is coded as:

```
      IF (.NOT. A) GO TO 200
        (code for condition TRUE)
      GO TO 300
 200  CONTINUE
        (code for condition FALSE)
 300  CONTINUE
```

Note: Code for IF condition is indented under the IF and code for the NOT IF is indented under the first continue.

B. DO-WHILE

This would be written as

```
  WHILE (A)
    (code)
  END WHILE
```

In FORTRAN, this is coded as:

```
1000  CONTINUE
      IF (.NOT.A) GO TO 2000
        (code)
      GO TO 1000
2000  CONTINUE
```

Note: Code to be done within a DO WHILE is indented under the IF statement. The CONTINUE statement in this case is indented as an END statement normally would be.

C. DO-UNTIL

This would be written as:

```
UNTIL (A)
  (code)
END UNTIL
```

In FORTRAN, this is coded as:

```
100   CONTINUE
        (code)
      IF (.NOT. A) GO TO 100
```

Note: Code to be done within a DO UNTIL is indented under the continue statement refined to beneath the loop. The IF statement is indented on the same level as the CONTINUE to which it refers.

D. CASE (Otherwise known as SELECT)

This would be written as:

```
SELECT (VALUE)
  When (ALT1)
    (code for alternative 1)
  When (ALT2)
    (code for alternative 2)
  When (ALTN)
    (code for alternative N)
  Otherwise
    (code for default)
END SELECT
```

In FORTRAN, this is implemented in one of two ways, depending on the type of values involved. It is implemented as a computed GO TO for consecutive numeric values, and as multiple IF statements, otherwise.

Computed GO TO (This example assumes 4 alternatives)

```
    GO TO (10,20,30,40), VALUE
      (code for default - value not between 1 and 4)
      GO TO 50
 10   CONTINUE
      (code for first alternative)
      GO TO 50
```

```
20   CONTINUE
     (code for second alternative)
     GO TO 50
30   CONTINUE
     (code for third alternative)
     GO TO 50
40   CONTINUE
     (code for fourth alternative)
50   CONTINUE
```

Multiple IF-Statements

(This example assumes 4 alternatives)

```
   IF (VALUE.NE.ALT1) GO TO 20
     (code for first alternative)
     GO TO 60
20 CONTINUE
   IF (VALUE.NE.ALT2) GO TO 30
     (code for second alternative)
     GO TO 60
30 CONTINUE
   IF (VALUE.NE.ALT3) GO TO 40
     (code for third alternative)
     GO TO 60
40 CONTINUE
   IF (VALUE.NE.ALT4) GO TO 50
     (code for fourth alternative)
     GO TO 60
50 CONTINUE
     (code when none of the alternatives are true)
60 CONTINUE
```

Note: Code for each alternative is coded under the previous statement. A CONTINUE statement is placed at the end of the FORTRAN CASE structure.

Logic Description - Program MAIN

(Module discussed in Paragraph 2.4.1)

```
BEGIN MAIN
Set Screen-Corners, Bed-Corners, Inch-Vertices to default values.
Set angle PHI and its trig function values to CALCOMP values.
CALL DEFALT - (Sets certain parameters to default values)
CALL HELP(1) -(Types opening message to user)
UNTIL (END)
  CALL ASK (1,REPLY) - (Types prompt "COMMAND?")
  Extract user response REPLY
  Select (REPLY)
    "PLOT" - PLOT is turned on
    "DUMP" - DUMP is turned on
    "END" - END is turned on, terminating the run.
    "?" - CALL HELPER - (Internal Subroutine)
    "BOX" - CALL BOX
    "ROTATE" - CALL ROTATE - (Internal Subroutine)
    "SCALE" - CALL SCALE - (Internal Subroutine)
    "FAST" - EXACT is turned off
    "EXACT" - EXACT is turned on
    "WINDOW" - CALL WINDOW
    "NOWINDOW" - BEGIN
      Set Screen-Corners, Bed-Corners, Inch-Vertices to default values
      Set angle PHI and its trig function values to CALCOMP values
      CALL DEFALT
      END
    "HELP" - CALL HELPER - (Internal Subroutine)
    "FRAME" - FRAME is turned on
    "NOFRAME" - FRAME is turned off
    "REORIGIN" - BEGIN
      HALT is turned on
      REORIGIN (RRIGIN) is turned on
      END
    "HALT" - HALT is turned on
    "NOHALT" - BEGIN
```

TABLE 2-01: Logic Description - Program MAIN
(Page 1 of 2)

```
        HALT is turned off
        REORIGIN (RRIGIN) is turned off
        END
      "FULLSCREEN" - BEGIN
        Set Screen-Corner, Bed-Corner, Inch-Vertices to fullscreen values
        Set angle PHI and its trig function values to Tektronix values
        CALL DEFALT
        END
      Otherwise - CALL HELP(17) - (Types "Invalid response")
      END Select (REPLY)
    WHILE (PLOT or DUMP)
      PLOTTING (PLOTNG) is turned on
      IF NOT (File-Open) THEN
        CALL FETCH(1) - Open File
        IF (PLOT) THEN
          CALL BLANKR (initializes the plotting process)
          ENDIF
        ENDIF
      WHILE (PLOTNG)
        STEP is turned off
        CALL FETCH(0) - (Retrieve Next CALCOMP Sentence)
        IF (PLOT) THEN
          CALL DRAW (Perform all graphic functions)
          ELSE
            Write out CALCOMP Sentence to UNIT 8
          ENDIF
        ENDWHILE (PLOTNG)
      PLOT is turned off
      DUMP is turned off
      ENDWHILE (PLOT or DUMP)
    ENDUNTIL (END)
  END MAIN
```

TABLE 2-01: Logic Description - Program MAIN

(Page 2 of 2)

Logic Description - Internal Subroutine SCALE

(Module discussed in Paragraph 2.4.1)

```
BEGIN SCALE
LOCFLG is turned off
UNTIL (LOCFLG)
  Analyze REPLY (Left over from user's input command)
  IF REPLY is numeric - THEN
    Set SCLRAT = REPLY
    LOCFLG is turned on
    ELSE
    SCLRAT is set to 0
    ENDIF
  IF SCLRAT ≤ 0 THEN
    Write "Invalid Reply"
    CALL ASK (3, REPLY) - (Prompt User to Enter Scale Factor)
    LOCFLG is turned off
    ENDIF
  ENDUNTIL
Divide CAL-Corners and Inch-Corners by SCLRAT
END SCALE
```

TABLE 2-02: Logic Description - Internal Subroutine SCALE

Logic Description - Internal Subroutine ROTATE

(Module discussed in Paragraph 2.4.1)

```
BEGIN ROTATE
LOCFLG is turned off
UNTIL (LOCFLG)
  Analyze REPLY (Left in from user's command)
  IF REPLY is numeric - THEN
    Compute PSI = REPLY/57.29578 - (Convert Degrees to Radians)
    Set COSPSI = COSINE (PSI)
    Set SINPSI = SINE (PSI)
    LOCFLG is turned on
    ELSE
    Write "Invalid Reply"
    CALL ASK (3, REPLY) - (Prompt user to Enter Angle in Degrees)
    ENDIF
  ENDUNTIL
END ROTATE
```

TABLE 2-03: Logic Description - Internal Subroutine ROTATE

Logic Description - Subroutine BLANKR
(Module discussed in Paragraph 2.4.2b)

```
BEGIN BLANKR
Set XTEK = Tek-Corners (3,1)
Set YTEK = Tek-Corners (3,2)
Set OLDX, OLDY, PREVX, PREVY = 1
Reorigin-Indicator (RRGIND) is turned off
Set CAL-To-Tek = Tek-Corners
DO:  K = 1 to 3 = (Create Work Matrix for use in transformation in LLSQ)
  CAL-Corners (K,1) = CAL-Corners (K,1) + XOFST
  CAL-Corners (K,2) = CAL-Corners (K,2) + YOFST
  Work-Matrix (K,1) = (CAL-Corners (K,1)*COSPSI) + (CAL-Corners (K,2)*SINPSI)
  Work-Matrix (K,2) = (CAL-Corners (K,2)*COSPSI) - (CAL-Corners (K,1)*SINPSI)
  Work-Matrix (K,3) = 1.0
  ENDDO
Set XOFST, YOFST = 0
CALL LLSQ - (Compute Transition matrix from CALCOMP coordinates to Tektronix)
IF (IER=0) THEN - (When Return from LLSQ okay)
  Set PSIX = PSI
  Set SINPSI = SINPSI
  Set COSPSX = COSPSI
  IF Default-On (DFLTON) THEN
  Tek-Corners = Screen-Corners
  CALL INITT - (Initialize terminal; Baud rate is a parameter)
  CALL TERM - (Takes advantage of features of Tektronix 4014/4015)
  CALL TWINDO (Tek-Corners (1,1), Tek-Corners (3,1), Tek-Corners (1,2),
    Tek-Corners (2,2)) - (Use upper and lower boundaries of terminal as parameters)
  CALL DWINDO (Tek-Corners (1,1), Tek-Corners (3,1), Tek-Corners (1,2)
    Tek-Corners (2,2))
  XMIN, YMIN = 67108864
  XMAX, YMAX = -XMIN
  Inch-To-Tek = Tek-Corners
  Do:  K = 1 to 3 - (Create Work Matrix for use in transformation in LLSQ)
```

TABLE 2-04: Logic Description - Subroutine BLANKR
(Page 1 of 3)

```
  Work-Matrix (K,1) = (Inch-Corners (K,1)*COSPSI) + (Inch-Corners (K,2)*SINPSI)
  Work-Matrix (K,2) = (Inch-Corners (K,2)*COSPSI) - (Inch-Corners (K,1)*SINPSI)
  Work-Matrix (K,3) = 1.0
  ENDDO
CALL LLSQ - (Convert Transition matrix from CALCOMP coordinates to Tektronix)
CALL MOVEA (Tek-Corners (1,1), Tek-Corners (1,2)) - Move beam to bottom left
        hand corner
IF (FRAME) THEN - (Draw Frame and Write Out Window Data)
  CALL DRAWA (Tek-Corners (1,1), Tek-Corners (2,2))
  CALL DRAWA (Tek-Corners (3,1), Tek-Corners (2,2))
  CALL DRAWA (Tek-Corners (3,1), Tek-Corners (1,2))
  CALL DRAWA (Tek-Corners (1,1), Tek-Corners (1,2))
  Write 'Window Data'
  CALL TRANS - (Using lower left hand corner - translate to inches)
  Write out values of coordinates for lower left hand corner.
  CALL TRANS - (Using upper right hand corner - translate to inches)
  Write out values of coordinates from upper right hand corner
  Compute SCALE = Square Root (ABS (Inch-To-Tek (1,1)*Inch-To-Tek (2,2)) -
                (Inch-To-Tek (2,1)*Inch-To-Tek (1,2))))
  Write out SCALE
  Write "Press Return to Begin Plot"
  CALL SCURSR (ICHAR, IX, IY) - (activate graphic cursor - holds display
             till user enters a character)
  CALL INITT (Reinitialize terminal)
  CALL TERM (Reset terminal characteristics)
  CALL TWINDO (Tek-Corners (1,1), Tek-Corners (3,1), Tek-Corners (1,2),
        Tek-Corners (2,2)) - (Enter upper and lower bounds of terminal)
  CALL DWINDO (Tek-Corners (1,1), Tek-Corners (3,1), Tek-Corners (1,2),
        Tek-Corners (2,2))
  (Draw Frame Before Actual Plot)
  CALL MOVEA (Tek-Corners (1,1), Tek-Corners (1,2))
  CALL DRAWA (Tek-Corners (1,1), Tek-Corners (2,2))
  CALL DRAWA (Tek-Corners (3,1), Tek-Corners (2,2))
  CALL DRAWA (Tek-Corners (3,1), Tek-Corners (1,2))
  CALL DRAWA (Tek-Corners (1,1), Tek-Corners (1,2))
```

TABLE 2-04: Logic Description - Subroutine BLANKR

(Page 2 of 3)

```
    ENDIF
  ELSE (Path if LLSQ fails - IER NOT EQUAL 0)
  Write 'Invalid Transformation Values'
  CALL FETCH(2) - (Close Plot File)
  Plotting (PLOTNG) is turned off
  ENDIF
END BLANKR
```

TABLE 2-04: Logic Description - Subroutine BLANKR
(Page 3 of 3)

Logic Description - Subroutine BOX

(Module discussed in Paragraph 2.4.2c)

```
BEGIN BOX
IF NOT (Reorigin-Indicator) THEN
  Set CAL-To-Tek(3,1) = XORG (Reset to X coordinate of original origin)
  Set CAL-To-Tek(3,2) = YORG (Reset to Y coordinate of original origin)
  Set XOFST, YOFST = 0
  ENDIF
Turn off PNT12, PNT13, PNT14, PNT23, PNT24, PNT34
Set X(I), Y(I) = 0 for I = 1,4
Set POINTS(1), POINTS(2) = 0
UNTIL (POINTS(2)> 0)
  CALL SCURSR (IX, IY, ICHAR) (Activate graphic cursor, obtain coordinates
           and designator)
  Compute ICHAR = ICHAR - 48 (Translate Machine Code to user response)
  SELECT (ICHAR)
    1,2,3,4 - BEGIN - (When the user has entered a valid first corner)
      Set X(ICHAR) = IX (Set X Coordinate)
      Set Y(ICHAR) = IY (Set Y Coordinate)
      POINTS(1) = ICHAR (Sets first corner)
      END
    Otherwise
      Write Error Message (Invalid designation of corner)
    END SELECT
***Below DO WHILE loop is not performed unless a valid first corner has been
   entered.***
  WHILE (POINTS(1)>0 and POINTS(2) <= 0) (Get second corner)
    CALL SCURSR (IX, IY, ICHAR)
    Set ICHAR = ICHAR - 48 - (Translate)
    IF ((ICHAR NE POINTS (1) and (ICHAR EQ 1,2,3,4) AND (IX NE X(1) OR
           IY NE Y(1)) THEN
      Set POINTS(2) = ICHAR
      Set X(ICHAR) = IX
      Set Y(ICHAR) = IY
      ENDIF
```

TABLE 2-05: Logic Description - Subroutine BOX

(Page 1 of 4)

```
      ELSE
        Write Error Message
        ENDIF
      ENDWHILE (POINTS(1) > 0 and POINTS(2) <= 0)
***If an invalid reply was received on the first try, the preceding WHILE loop
   is not executed, and control passes here, which executes the outer loop again.***
     ENDUNTIL (POINTS(2) > 0)
  Select (POINTS(1) and POINTS(2)) -(Select Desired Corners)
    Corners = 1,2 - BEGIN
      RHO2 = Square Root ((X(2) - X(1))^2 + (Y(2) - Y(1))^2)
      RHO3 = RHO2 * CSCPHI
      RHO4 = RHO2 * COTPHI
      THETA2 = ARCTAN ((Y(2)-Y(1))/(X(2)-X(1)))
      THETA4 = THETA2 - (π/2)
      THETA3 = THETA4 + PHI
      X(3) = RHO3 * COS(THETA3) + X(1)
      Y(3) = RHO4 * SIN(THETA3) + Y(1)
      X(4) = RHO4 * COS(THETA4) + X(1)
      Y(4) = RHO4 * SIN(THETA4) + Y(1)
      END
    Corners = 1,3 - BEGIN
      RHO3 = Square Root ((X(3)-X(1))^2 + (Y(3) - Y(1))^2)
      RHO2 = RHO3 * SINPHI
      RHO4 = RHO3 * COSPHI
      THETA3 = ARCTAN ((Y(3)-Y(1))/X(3)-X(1)))
      THETA4 = THETA3 - PHI
      THETA2 = THETA4 + (π/2)
      X(2) = RHO2 * COS(THETA2) + X(1)
      Y(2) = RHO2 * SIN(THETA2) + Y(1)
      X(4) = RHO4 * COS(THETA4) + X(1)
      Y(4) = RHO4 * SIN(THETA4) + Y(1)
      END
```

TABLE 2-05: Logic Description - Subroutine BOX

(Page 2 of 4)

```
Corners = 1,4 - BEGIN
  RHO4 = Square Root ((X(4)-X(1))^2 + (Y(4) - Y(1))^2)
  RHO2 = RHO4 * TANPHI
  RHO3 = RHO4 * SECPHI
  THETA4 = ARCTAN ((Y(4)-Y(1))/(X(4)-X(1)))
  THETA3 = THETA + PHI
  THETA2 = THETA4 + (π/2)
  X(3) = RHO3 * COS(THETA3) + Y(1)
  Y(3) = RHO3 * SIN(THETA3) + Y(1)
  X(2) = RHO2 * COS(THETA2) + X(1)
  Y(2) = RHO2 * SIN(THETA2) + Y(1)
  END
Corners = 2,3 - BEGIN
  RHO4 = Square Root ((X(3)-X(2))^2 + (Y(3) - Y(2))^2)
  RHO2 = RHO4 * TANPHI
  RHO3 = RHO4 * SECPHI
  THETA4 = ARCTAN ((Y(3)-Y(2))/(X(3) - X(2)))
  THETA3 = THETA4 + PHI
  THETA2 = THETA4 + (π/2)
  X(1) = X(2) - COS(THETA2) * RHO2
  Y(1) = Y(2) - SIN(THETA2) * RHO2
  X(4) = RHO4 * COS(THETA4) + X(1)
  Y(4) = RHO4 * SIN(THETA4) + Y(1)
  END
Corners = 4,2 - BEGIN
  RHO2 = Square Root ((X(3)-X(4))^2 + (Y(2) - Y(4))^2)
  RHO3 = RHO2 * CSCPHI
  RHO4 = RHO2 * COTPHI
  THETA2 = ARCTAN ((Y(3)-Y(4))/(X(3) - X(4)))
  THETA4 = THETA2 - (π/2)
  THETA3 = THETA4 + PHI
```

TABLE 2-05 Logic Description - Subroutine BOX

(Page 3 of 4)

```
    X(1) = X(4) - COS(THETA4) * RHO4
    X(1) = Y(4) - SIN(THETA) * RHO4
    X(2) = RHO2 * COS(THETA2) + X(1)
    Y(2) = RHO2 * SIN(THETA2) + Y(1)
    END
  END Select
CALL MOVEA (X(1), Y(1)) - (Move to lower left hand corner of Box)
DO:  J = 1 TO 3 - (DO transformations and Draw Sides of Box)
  Set U1 = X(J)
  Set U2 = Y(J)
  CALL TRANS (U1, U2, CAL-To-Tek)
  Compute:
    CAL-Corners (J,1) = (U1 * COSPSX) - (U2 * SINPSX)
    CAL-Corners (J,2) = (U1 * SINPSX) + (U2 * COSPSX)
  CALL DRAWA (X(J + 1), Y(J + 1)) - (Draw Side of Box),
  ENDDO
CALL DRAWA (X(1), Y(1)) - (Draw last side of Box)
CALL SCURSR (ICHAR, IX, IY) - (Dump Buffer by displaying screen cursor)
Set Tek-Corners = Screen-Corners
END BOX
```

TABLE 2-05: Logic Description - Subroutine BOX

(Page 4 of 4)

Logic Description - Subroutine CALTEK

(Module discussed in Paragraph 2.4.2d)

```
BEGIN CALTEK
Set XBFORE = XPRIME
Set YBFORE = YPRIME
Set OLDX = XTEK
Set OLDY = YTEK
(Apply transformations to XCAL and YCAL to result in XPRIME and YPRIME,
the absolute CALCOMP screen coordinate in floating point)
XPRIME = ((XCAL - UVALUE) * PVALUE) +
         ((YCAL - VVALUE) * QVALUE)
YPRIME = ((XCAL - UVALUE) * RVALUE) +
         ((YCAL - VVALUE) * SVALUE)
(Apply locally specified transformations to derive the absolute
Tektronix screen coordinates)
XTEK = CAL-To-Tek (1,1) * XPRIME +
       CAL-To-Tek (2,1) * YPRIME + CAL-To-Tek (3,1)
YTEK = CAL-To-Tek (1,2) * XPRIME +
       CAL-To-Tek (2,2) * YPRIME + CAL-To-Tek (3,2)
Set PREVX = XTEK
Set PREVY = YTEK
STEP is turned on
***GVALUE = 52 means plot a string of ASCII characters.  GVALUE = 53 means
   plot a single 'special character'.***
IF (Pen-Down or GVALUE = 52 or GVALUE = 53) THEN
  IF XPRIME > XMAX Set XMAX = XPRIME
  IF XPRIME < XMIN Set XMIN = XPRIME
  IF YPRIME > YMAX Set YMAX = YPRIME
  IF YPRIME < YMIN Set YMIN = YPRIME
  ENDIF
END CALTEK
```

TABLE 2-06: Logic Description - Subroutine CALTEK

Logic Description - Subroutine CIRCLE

(Module discussed in Paragraph 2.4.2e)

```
BEGIN CIRCLE
(Locate Center of Circle)
XCENTR = XBFORE + IVALUE
YCENTR = YBFORE + JVALUE
***Determine whether to draw full or partial circle.***
Q1 = (IVALUE)^2 + (JVALUE)^2
Q2 = (XCAL - XCENTR)^2 + (YCAL - YCENTR)^2
STANG = ARCTAN (-JVALUE/-IVALUE)
ENDANG = ARCTAN ((YCAL - YCENTR)/XCAL - XCENTR))
IF (GVALUE = 2) THEN (Clockwise Circke)
  IF (Q1 NOT = Q2) THEN (Full Circle)
    ENDANG = STANG - 2
    ELSE
  IF ENDANG > STANG THEN - (Partial Circle)
    ENDANG = ENDANG - 2
    ENDIF
  Set SIGN = -1
  ELSE (Counterclockwise Circle)
  IF (Q1 NOT = Q2) THEN (Full Circle)
    ENDANG = STANG + 2
  ELSE
  IF ENDANG < STANG THEN - (Partial Circle)
    ENDANG = ENDANG + 2
    ENDIF
  Set SIGN = +1
  ENDIF
***Compute value of angular increment.***
IF (EXACT) THEN (Compute smallest angular change useable to give precise circle)
  Compute DET = (CAL-To-Tek(1,1) * CAL-To-Tek(2,2)) -
    (CAL-To-Tek(1,2) * CAL-To-Tek(2,1))
  Compute DET = DET * ((PVALUE * SVALUE) - (QVALUE * RVALUE))
```

TABLE 2-07: Logic Description - Subroutine CIRCLE

(Page 1 of 2)

```
  IF (DET = 0) THEN
    RETURN
    ELSE
    Compute DELTA = SIGN/(2π* SQUARE ROOT (Q1 * ABS (DET)))
    ENDIF
    ELSE (Approximate circle by polygon)
    Compute DELTA = (ENDANG - STANG)/32
    ENDIF
  IF (DELTA = 0) DELTA = 3 * SIGN
  Set Q1 = Square Root (Q1)
  Set:
  Work-Matrix (1,1) = PVALUE
  Work-Matrix (1,2) = QVALUE
  Work-Matrix (1,3) = UVALUE
  Work-Matrix (2,1) = RVALUE
  Work-Matrix (2,2) = SVALUE
  Work-Matrix (2,3) = VVALUE
  CALL TRANS (XCENTR, YCENTR, Work-Matrix)
  DO:  ANGLE = STANG TO ENDANG BY DELTA (Draw the Circle)
    XCAL = XCENTR + COS(ANGLE) * Q1
    YCAL = YCENTR + SIN(ANGLE) * Q1
    STEP is turned off
    CALL CALTEK
    IF (STEP) THEN CALL DRAW (XTEK, YTEK)
    ENDDO
END CIRCLE
```

TABLE 2-07: Logic Description - Subroutine CIRCLE
(Page 2 of 2)

Logic Description - Subroutine DECODE

(Module discussed in Paragraph 2.4.2f)

```
BEGIN DECODE
End-Found is turned off (End of sentence not yet found)
Set PNTR = 1 (PNTR points to first character in sentence)
UNTIL (End-Found)
  Set PNTRHD = PNTR (Save position of current letter code)
  Increment PNTR
  Find next letter code; i.e. Set NEXT = to position of first non-numeric
        character at or after the PNTR 'th position
  IF (NEXT = PNTR) THEN
    Set INTVAL = 0 (When letter code is followed by letter code)
    ELSE (Extract integer value from sentence)
    Set INTVAL = Numeric characters in between the PNTR 'th
           and NEXT 'th positions in sentence
    Set PNTR = NEXT (Set to position of next letter code)
    ENDIF
  Select (Current letter code) - (At PNTRHD 'th position)
    "X" - Set XCAL = INTVAL
    "Y" - Set YCAL = INTVAL
    "E" - Set EVALUE = INTVAL
    "F" - Set FVALUE = INTVAL
    "I" - Set IVALUE = INTVAL
    "J" - Set JVALUE = INTVAL
    "P" - Set PVALUE = INTVAL/10000.
    "Q" - Set QVALUE = INTVAL/10000.
    "R" - Set RVALUE = INTVAL/10000.
    "S" - Set SVALUE = INTVAL/10000.
    "A" - Set AVALUE = INTVAL
    "B" - Set BVALUE = INTVAL
    "U" - Set UVALUE = INTVAL
    "V" - Set VVALUE = INTVAL
```

TABLE 2-08: Logic Description - Subroutine DECODE

(Page 1 of 3)

```
    "M" - DO
      Set MVALUE = INTVAL
      STEP is turned on
      ENDDO
    "T" = Set TVALUE = INTVAL
    "G" - Perform G-Code Processing (Handle character strings - below)
    "D" - BEGIN
    IF (GVALUE NOT = 50) THEN
***GVALUE of 50 means to select the pen number.  In this case, the pen number
   is contained in the D code.  Pen number is not significant to this program.
   Otherwise, for any other value of GVALUE, D1 means pen down, D2 means
   pen up.***
    Select (INTVAL)
      "1" - turn on PNDOWN
      "2" - turn off PNDOWN
      END Select
    END
 END Select
IF Current position of sentence is a period turn on End-Found
ENDUNTIL
END DECODE
```

TABLE 2-08: Logic Description - Subroutine DECODE

(Page 2 of 3)

```
BEGIN G-Code Process
Select (INTVAL)
  52, 55 - BEGIN
    Increment PNTR
    Set NEXT to position of exclamation point that delimits character string
    Set LVALUE = NEXT - PNTR
    Set PNTR = NEXT + 1
    Set NVALUE = 1.875
    END
  53 - BEGIN - (Distinguish centered characters from regular symbols,
               based on value of SPCIL)
    Set PNTR = PNTR + 3
    IF (SPCIL≥ 15) THEN (Centered symbols)
      NVALUE = 1
    ELSE
      NVALUE = 1.875
      ENDIF
      ENDDO
  5 - BEGIN (Set existing '5' code back to zero, so that code interpreting
            sentences know whether it is at beginning or end of spline
            interpolation task.)
    IF GVALUE = 5
    Set INTVAL = 0
    END
  END Select
Set GVALUE = INTVAL (Done for all G-codes)
END G-Code Process
```

TABLE 2-08: Logic Description - Subroutine DECODE
(Page 3 of 3)

Logic Description - Subroutine DEFALT

(Module discussed in Paragraph 2.4.2g)

```
BEGIN DEFALT
Default-On is turned on
Set AVALUE, BVALUE = 0.5
CALL OPMSG(0) - (Set number of lines on terminal to 720)
Set GVALUE = 0
Set SINPSI, PSI, QVALUE, RVALUE, UVALUE, VVALUE = 0
Set PVALUE, SVALUE, COSPSI = 1
Set Cal-Corners = Bed-Corners
Set Tek-Corners = Screen-Corners
Set Inch-Corners = Inch-Vertices
END DEFALT
```

TABLE 2-09: Logic Description - Subroutine DEFALT

Logic Description - Subroutine DRAW

(Module discussed in Paragraph 2.4.2h)

```
BEGIN DRAW
CALL DECODE - (Interpret Next CALCOMP Sentence)
CALL CALTEK - (Apply all translations and rotations to coordinates;
               Compute new values of XTEK and YTEK)
Select (MVALUE)
  1 - BEGIN (Temporary halt)
    Set MVALUE = 0
    IF (HALT) THEN
      Write out blank line
      CALL SCURSR - (Pause for Instructions)
      CALL MOVEA (XTEK, YTEK) = (Return to Position)
      Select (Character Entered)
      'P' - PLOTTING is turned off
      '.' - CALL WRAPUP
      OTHERWISE NO ACTION TAKEN
      END Select
    ENDIF
  END
2 - BEGIN (Final halt)
  Set MVALUE = 0
  CALL WRAPUP
  END
```

TABLE 2-10: Logic Description - Subroutine DRAW

(Page 1 of 3)

```
OTHERWISE (Path for normal plotting actions)
  SELECT (GVALUE)
    1 - BEGIN (Draw solid line or move)
      IF (STEP) THEN
        IF (Pen-Down) THEN CALL DRAWA (XTEK, YTEK)
        ELSE CALL MOVEA (XTEK, YTEK)
      END
    END
    4 - BEGIN (Draw dashed line or move)
      IF (STEP) THEN
        IF (Pen-Down) THEN CALL DASHA (XTEK, YTEK, 78)
        ELSE CALL MOVEA (XTEK, YTEK)
      END
    END
      2,3 - CALL CIRCLE - (Draw Circles)
      5 - CALL SPLINE (1) - (Begin Spline Interpolation)
      53 - CALL SPECL - (draw Special Characters)
      52 - CALL STNDRAD - (draw Standard Characters)
      55 - CALL OPMSG(1) - (Write Operator Messages)
      0 - (Terminate Spline Interpolation)
      IF (CURVE) CALL SPLINE (0)
      END
    END SELECT
  END (Otherwise path)
END Select
```

TABLE 2-10: Logic Description - Subroutine DRAW
(page 2 of 3)

```
***GVALUE = 25 means reorigin to current pen location.***
 IF (GVALUE = 25) THEN (Perform reorigin of coordinates)
   IF (Reorigin-Indicator not yet turned on) THEN
     Reorigin-Indicator is turned on
     Set XOFST = XBFORE (To translate plot to correct position)
     Set YOFST = YBFORE
     Set X-Origin = CLTTK(3,1) - (Save old origin)
     Set Y-Origin = CLTTK(3,2)
     ENDIF
   IF (RRIGIN) THEN - (Save Screen Cursor as origin of plot)
     CALL SCURSR (IX, IY, ICHAR)
     Set CAL-To-Tek(3,1) = IX
     Set CAL-To-Tek(3,2) = IY
   ELSE - (Reset origin to previous origin)
     Set CAL-To-Tek(3,1) = OLDX
     Set CAL-To-Tek(3,2) = OLDY
     ENDIF
   Set GVALUE, XCAL, YCAL = 0
   ENDIF
ENDDRAW
```

TABLE 2-10: Logic Description - Subroutine DRAW

(Page 3 of 3)

Logic Description - Internal Subroutine WRAPUP

(Module discussed in Paragraph 2.4.2h)

```
BEGIN WRAPUP
CALL FETCH(2) - (Close Plot File)
CALL DEFALT - (Reset default values)
CALL SCURSR - (Hold plot until user finishes looking at it)
Plotting is turned off.
(Prepare for optimization by computing XRATIO and YRATIO)
Compute:
  XRATIO = (XMAX - XMIN)/Bed-Corners (3,1))
  YRATIO = (YMAX - YMIN)/243840
IF (XRATIO > YRATIO) THEN
  COMPUTE YMAX = YMIN + (243840 * XRATIO)
  ELSE
  COMPUTE XMAX = XMIN + (Bed-Corners(3,1) * YRATIO)
  ENDIF
Set lower left hand corner of CAL-Corners to XMIN and YMIN
Set upper left hand corner of CAL-Corners to XMIN and YMAX
Set upper right hand corner of CAL-Corners to XMAX and YMAX
Set lower left hand corner of Tek-Corners to 5 and Screen-Corners + 4
Set upper left hand corner of Tek-Corners to 5 and 3117
Set upper right hand corner in Tek-Corners to 4090 and 3117
Compute Inch-Corners = CAL-Corners/5080
Write "Final Halt"
END WRAPUP
```

TABLE 2-11: Logic Description - Subroutine WRAPUP

Logic Description - Subroutine FETCH
(Module discussed in Paragraph 2.4.2i)

```
BEGIN FETCH (IFUNCT) (May be 0,1,2)
Select (IFUNCT)
  0 - Extract next CALCOMP sentence
  1 - Prepare for first input record
  2 - Terminate file processing
  Otherwise - STOP 1040 (Error Condition)
  END Select
RETURN
END FETCH

BEGIN EXTRACT (next CALCOMP sentence; when IFUNCT .EQ. 0)
Set Special-length to 0
Set Build-start pointer to 1
Set truncated, sentence-completed, and in-string to .FALSE.
UNTIL (Sentence-completed or Truncated)
  WHILE (Record-Needed)
    (This WHILE group controls the input of the physical records.  The
    CALCOMP sentences are variable length, and may be spanned over
    physical records.)
    Read File INFILE (at end of file, terminate file processing) into
      Record-IN
    Set Translated-record to Record-in
    Call TRNSLT (TRNRCD, length of TRNRCD, 1)
      (Translate special CALCOMP ASCII characters to EBCDIC (NASC AS/5)
       or ASCII (PDP-11)
    IF ("SYNC-Code" character string is in proper location, verifying
      that this is a correct CALCOMP record) THEN
      Set Record-needed to .FALSE.
      Set Scan-start-pointer to 1
      ENDIF
    ENDWHILE (Record-needed)
```

TABLE 2-12: Logic Description - Subroutine FETCH
(Page 1 of 4)

```
IF (In-String) THEN
  Set J to (remaining length of data portion of translated record,
    in bytes)
  Set Work-record to remaining data in translated record.
  Set TERMINATOR to position of the delemiter following the quoted
    character string.
  Place the untranslated data corresponding to the quoted character
    string into the array Special-Text, and note the length of the
    string.
  Set In-string to .FALSE.
  Adjust Build-Start, Scan-Start, and Special length.
    IF (Scan-start is beyond the end of the record) THEN
    Set Record-needed to .TRUE.
    ELSE
    IF (Next character to be scanned is the end-of-data marker ($))
    Set Record-needed to .TRUE.
    ENDIF
  ELSE (In-String is .FALSE.)
  (This is the path taken at the beginning of a sentence, or when a
  sentence is continued over two physical records.)
  Set J to (remaining length of data portion of translated record,
    in bytes.)
  Extract last J bytes of translated record into Work-record.
  (Locate the delimiter that governs the action to be taken:  It is the
    first of these three characters to occur in the work-record:
      '.' marks the end of a sentence.
      '!' marks the beginning and end of a string of characters.
      '$' marks the end of the data portion of the record.
  Neither '.' nor '$' is significant when enclosed within paired
    exclamation marks.
  The only way the exclamation mark character may be transmitted
    is in a string of length 1, as '!!!'.)
```

TABLE 2-12: Logic Description - Subroutine FETCH

(Page 2 of 4)

```
Select
  WHEN (there is no break ('$')) STOP 1041
       (This is a defective CALCOMP record)
    WHEN ('!' occurs first)
      BEGIN (Open a character string)
        IF (character string is of the form '! X !', where X is the value)
          THEN
          Place untranslated byte corresponding to X into low-order
          byte of SPCIL.  (How this is done differs between NASC
          and PDP; for NASC, divide by 16777216 to shift right -
          PDP, subtract value of high-order blanks.)
          Adjust length of sentence to be extracted.
          ELSE
          Set In-string to .TRUE.
          Adjust length of sentence to be extracted.
          ENDIF
        END (Open a character string)
    WHEN ('!' occurs first)
      BEGIN (Complete the sentence)
      Set Sentence-completed to .TRUE.
      Adjust length to be extracted.
      IF (Next character to be scanned is '$') THEN
        Set Record-needed to .TRUE.
        (Next request for a sentence will cause a physical record to be
        brought in)
        ENDIF
      END (Complete the sentence)
    WHEN ('$' occurs first)
      BEGIN (Extract partial sentence)
        Adjust length to be extracted.
        Set Record-needed = .TRUE.
        END (Extract Partial sentence)
  END Select
```

TABLE 2-12: Logic Description - Subroutine FETCH
(Page 3 of 4)

```
    IF (Length of completed sentence would exceed length of SNTNCE) THEN
      Set Truncated = .TRUE.
      Adjust length to maximum possible of additional bytes.
      ENDIF
    IF (Length to be extracted .GT. 0) THEN
      EXTRACT appropriate number of bytes into Work-sentence
      Adjust build-start
      Adjust scan-start
      ENDIF
    IF (Truncated or sentence-completed) THEN
      Compute length of sentence (LENSNT)
      Place first (LENSNT) bytes of work sentence into SNTNCE
      IF (Truncated)
      Insert a period ('.') into last byte of SNTNCE.
      ENDIF
  (At this point, if Truncated or sentence-completed, then exit from
  the program.  If not, then control returns to the top of the loop to
  complete the sentence.)
  ENDUNTIL (Sentence-completed or truncated)
END EXTRACT

BEGIN PREPARE (for first input record - IFUNC .EQ. 1)
Set Record-needed to .TRUE.
Set File-open to .TRUE.
(For PDP-11 version only:
Call GETFIL to construct file name and open the input file.)
END PREPARE

BEGIN TERMINATE (file processing)
REWIND INFILE
Set FILOPN = .FALSE.
(For PDP-11 version only:  CALL GETFIL to close file.)
END TERMINATE
```

TABLE 2-12: Logic Description - Subroutine FETCH

(Page 4 of 4)

Logic Description - Subroutine OPMSG

(Module discussed in Paragraph 2.4.2k)

```
BEGIN OPMSG
IF (FUNCT = 0) THEN
Set LINE = 720 - (Reset initial conditions)
  ELSE - (Put the message on the terminal and get results)
    Select (LVALUE) (Adjust character size according to length of message)
    LVALUE≤ 0 Return
    LVALUE≤ 70 BEGIN
      Set SIZE = 1
      Set LINES = 1
      Set POINTS = 68
      END
    LVALUE≤ 77 BEGIN
      Set SIZE = 2
      Set LINES = 1
      Set POINTS = 83
      END
    LVALUE≤ 117 BEGIN
      Set SIZE = 3
      Set LINES = 1
      Set POINTS = 53
      END
    LVALUE≤129 BEGIN
      Set SIZE = 4
      Set LINES = 1
      Set POINTS = 53
    Otherwise BEGIN
      Set SIZE = 4
      Compute LINES + (Length (special text) +4)/113 + 1
      Set POINTS = 53
      END
    END Select
```

TABLE 2-13: Logic Description - Subroutine OPMSG

(Page 1 of 2)

OPMSG attempts to keep all operator messages in the reserved space below the graphics area on the CRT. If that space is filled up, or if the program is in fullscreen mode, it writes operator messages from the top of the CRT down.

```
IF (LINE ≤POINTS * LINES) THEN
  Set LINE = 3119 - POINTS
  ELSE
  Set LINE = LINE - POINTS
  ENDIF
CALL TWINDO (0,4095, 0, 3120) - (For Entire Tektronix Screen)
CALL DWINDO (0,4095, 0, 3120)
CALL CHRSIZ (SIZE)
Translate operator message from machine code to text to be displayed;
Place 3 asterisks in front of message.
CALL MOVEA (0, LINE) - (Move below plot)
CALL AOUTST (LVALUE + 4, Message) - (Display operator message)
LINE = LINE - POINTS * (LINES -1)
CALL SCURSR (ICHAR, IX, IY) (Display graphic cursor)
IF (Reorigin) THEN (Set origin to coordinates of cursor)
CAL-To-Tek (3,1) = IX
CAL-To-Tek (3,2) = IY
ENDIF
CALL MOVEA (XTEK, YTEK)
CALL TWINDO - (For Tektronix Corners)
CALL DWINDO - (For Tektronix Corners)
ENDIF
END OPMSG
```

TABLE 2-13: Logic Description - Subroutine OPMSG
(Page 2 of 2)

Logic Description - Subroutine SPECL

(Module discussed in Paragraph 2.4.21)

```
BEGIN SPECL
CALL SYMHGT - (Establish orientation of special character)
IF any values out of range, THEN
RETURN
ELSE
  DO:  J=POINT (Special Character) TO POINT (Next Character) -1 BY 2
    MOVE is turned off
    Select (Entry(J))
      (ENTRY(J) ≤ -100) BEGIN
        MOVE is turned on
        Compute DELTAX = (ENTRY(J) +100) * HEIGHT
        END
      (ENTRY(J) ≥ 100) BEGIN
        MOVE is turned on
        Compute DELTAX = (ENTRY(J) -100) * HEIGHT
        END
      Otherwise BEGIN
        MOVE is turned on
        Compute DELTAX = ENTRY(J) * HEIGHT
        END
      END Select
  XCAL = XCAL + (DELTAX * CSTHET) - (DELTAY * SNTHET)
  YCAL = YCAL + (DELTAY * SNTHET) + (DELTAY * CSTHET)
  CALL CALTEK - (Calculate XTEK and YTEK from new values of XCAL and YCAL)
  IF MOVE THEN
    CALL MOVEA (XTEK, YTEK)
    ELSE
    CALL DRAWA (XTEK, YTEK)
    ENDIF
  ENDDO
ENDIF
END SPECL
```

TABLE 2-14: Logic Description - Subroutine SPECL

Logic Description - Subroutine SPLINE

(Module discussed in Paragraph 2.4.2m)

```
BEGIN SPLINE
IF (FUNCT = 0) - THEN - (End of spline interpolation)
  CURVE is turned off
  Set RUNX, RUNY = 0
  Set COUNT = 0
  ELSE
  Increment COUNT
  CURVE is turned on
  Set RUNX(COUNT), RUNY(COUNT) to XTEK, YTEK
  (coordinates of current point)
  IF (COUNT = 3) THEN
    CALL MOVEA (XTEK, YTEK) - (Positions the plotter beam at the
    starting point for the curve to be plotted)
    ENDIF
  IF (COUNT = 5) THEN
    Set each RUNX(J) and RUNY(J) to RUNX(J) and RUNY(J-1),
    skipping over previously used phantom point
    IF (EXACT) THEN - (Perform interpolation)
      Set X1 = RUNX(2)
      Set Y1 = RUNY(2)
      Set X2 = RUNX(3)
      Set Y2 = RUNY(3)
      Compute DELTA = 2/Maximum of (X2-X1) and (Y2-Y1)
      Select:
        WHEN (First two members of RUNX, RUNY are euqal) - BEGIN
          Set quadratic coefficients to last three members of RUNX and RUNY
          QUADRATIC is turned on
          CALL LLSQ using parameter matrix T2MAT and Quadratic Coefficients
          END
        WHEN (Third and fourth members of RUNX, RUNY are equal) - BEGIN
          Set quadratic coefficients to first three members of RUNX and RUNY
          QUADRATIC is turned on
          CALL LLSQ using parameter matrix T3MAT and Quadratic Coefficients
          END
```

TABLE 2-15: Logic Description - Subroutine SPLINE

(Page 1 of 2)

```
        Otherwise - BEGIN - (Cubic interpolation)
          Set cubic coefficients to first four members of RUNX and RUNY
          CALL LLSQ using parameter matrix T4MAT and cubic coefficients
          Set XCOEFF and YCOEFF to current values of Cubic Coefficients
          DO:  TX from -1 to 1 by DELTA - (Plot curve)
            Compute XCORD and YCORD to find the end point of the small
            line segment used to approximate curve
            CALL DRAWA (XCORD, YCORD)
            ENDDO
          END
        END Select
      IF (Quadratic) - (Quadratic Interpolation) THEN
        Set first three members of XCOEFF and YCOEFF to current values
        of quadratic coefficients
          DO:  TX from -1 to 1 by DELTA - (Plot curve)
            Compute XCORD and YCORD using TX, XCOEFF, and YCOEFF to find
            the end point of the small line segment used to approximate
            curve
            CALL DRAW (XCORD, YCORD)
            ENDDO
          ENDIF
        ENDIF
      ENDIF
    ELSE - (Approximate curve between two points with line segment)
       CALL DRAW (RUNX(3), RUNY(3))
         ENDIF
    COUNT = 4
    ENDIF
  ENDIF
END SPLINE
```

TABLE 2-15: Logic Description - Subroutine SPLINE
(Page 2 of 2)

Logic Description - Subroutine STNDRD

(Module discussed in Paragraph 2.4.2n)

```
BEGIN STNDRD
CALL SYMHGT - (Calculates orientation of character)
IF any parameters are invalid THEN
  RETURN
  ELSE
  DO:  I = 1 to LVALUE - (Do until last character drawn)
    K1 = Internal representation of character, converted to actual
         number used as pointer
    IF (K1 is between 0 and 65) THEN
      DO J = PNTR (K1) to PNTR (K1 +1) -1 BY 2:
        Select (ENTRY(J))
          (ENTRY(J) ≤ -100) BEGIN
            MOVE is turned on
            Compute DELTAX = (ENTRY(J) +100) * HEIGHT
            END
          (ENTRY(J) ≥ 100) BEGIN
            MOVE is turned on
            Compute DELTAX = (ENTRY(J) -100) * HEIGHT
            END
          Otherwise BEGIN
            MOVE is turned off
            Compute DELTAX = ENTRY(J) * HEIGHT
            END
          END Select
        XCAL = XCAL + (DELTAX * CSTHET) -
                      (DELTAY * SNTHET)
        YCAL = YCAL + (DELTAX * SNTHET) +
                      (DELTAY * CSTHET)
        CALL CALTEK - (Convert to Tektronix coordinates)
        IF (MOVE) THEN CALL MOVEA (XTEK, YTEK)
        ELSE CALL DRAWA (XTEK, YTEK)
        ENDIF
      ENDDO
    ENDDO
  ENDIF
END STNDRD
```

TABLE 2-16: Logic Description - Subroutine STNDRD

Logic Description - Subroutine WINDOW

(Module discussed in Paragraph 2.4.2o)

```
BEGIN WINDOW
Default-On is turned on
DO:  J=4,7 - (Since number 4-7 in ASK contain window prompts)
  FLAG is turned off
  UNTIL (FLAG) - (Valid value entered)
    CALL ASK (J, REPLY) - (Prompt for window value)
    IF REPLY is numeric THEN
      VALUE (J-3) = REPLY
      FLAG is turned on
      ELSE
      Write error message
      ENDIF
    ENDUNTIL
  ENDDO
Set lower left hand corner in CAL-Corners = X1, Y1
Set upper left hand corner in CAL-Corners = X1, Y2
Set upper right hand corner in CAL-Corners = X2, Y2
Set Inch-Corners = CAL-Corners
IF (X1 = X2) RETURN
RATIO = Absolute value of (Y2-Y1)/(X2-X1)
IF (RATIO=0) RETURN
IF (RATIO ≤ (Bed-Corners (3,2)/Bed-Corners (3,1))) THEN
  Set Tek-Corners (2,2), Tek-Corners (3,2) = Screen-Corners (2,2)
  Compute Tek-Corners (3,1) = (Screen-Corners (2,2) - Screen-Corners (1,2)/RATIO
  ELSE
  Set Tek-Corners (3,1) = Screen-Corners (3,1)
  Compute Tek-Corners (2,2) and Tek-Corners (3,2) =
    Screen-Corners (1,2) + (RATIO * Screen-Corners (3,1))
  ENDIF
```

TABLE 2-17: Logic Description - Subroutine WINDOW

(Page 1 of 2)

Set Tek-Corners (1,1), Tek-Corners (2,1) = 0

Set Tek-Corners (1,2) = Screen-Corners (1,2)

Set PSI = -PSI

Set SINPSI = -SINPSI

END WINDOW

TABLE 2-17: Logic Description - Subroutine WINDOW
(Page 2 of 2)

AD-A102 493 PRC DATA SERVICES CO RESTON VA F/G 9/2
CALCOMP PREVIEW SYSTEM. PROGRAM MAINTENANCE MANUAL.(U)
APR 81 DCA100-77-C-0037
UNCLASSIFIED DCA/DF-81/002A NL

2 OF 2
AD A
102493

END
DATE
FILMED
8-81
DTIC

SECTION 3. ENVIRONMENT

3.1 Equipment Environment. On the NASC AS/5-3, the CALCOMP Preview System requires the following to run: a CPU with 192K of storage, a TEKTRONIX 4014-1 terminal or equivalent for input and output, and a CALCOMP file on disk. CALCOMP files on tape can be copied to disk through use of the utility IEBGENER. See Reference 1.2e for details. A user merely wishing a dump of the file can run the job in batch and can use the line printer for output.

On the PDP-11/70, the CALCOMP Preview System needs the following to run: a CPU, a TEKTRONIX 4014-1 terminal for input and output, and a CALCOMP file on either tape or disk.

3.2 Support Software. On the NASC AS/5-3, the programs were compiled using the FORTRAN H Extended Compiler - Optimization Level 1. The string handling routines were developed using IBM 360 ALC. On the PDP-11/70, the programs were compiled using FORTRAN 4 PLUS V03.0.

SECTION 4. PROGRAM MAINTENANCE PROCEDURES

4.1 Retrieval Procedures. The source code of the program described in Sections 2.4.1 - 2.4.4 is stored in R831.LIBRARY under the member name CLTOTK. It can be retrieved through the LBGET command. For example:

LBGET CLTOTK DSSOURCE('R831.LIBRARY') CNTL

The load module is stored in SYS9.DCAFORT.LINKLIB under the number name TEKTRONX. During maintenance and modification, the source code and load module should be left unchanged. Instead, changes should be made to copies of the subroutines. If extensive changes are to be made, the recommendation is to divide the program source into separate modules, recompile each module with the option of saving the object code, and link edit. See Figure 4-01 for sample JCL for assembling a string handling routine. See Figure 4-02 for sample JCL for compilation and linkage edit.

The PDP-11 version of the CALCOMP Preview program can be retrieved in one of two ways. One way, the user LOGIN's to the PDP-11 and copies the program from UIC [50,55] to his own UIC. He types in:

COPY [50,55]CLTOTK.FTN CLTOTK.FTN

The user now has a copy of CLTOTK.FTN under his UIC.

Alternatively, the user can do the following. The user obtains a tape from the room containing the PDP-11/70 and gives it to the operators of the NASC who give out a serial number for use on the NASC. Then the user logs on to the NASC, performs a LBGET of CLTKPDP from R831.LIBRARY, places CLTKPDP on the tape through the ITP command, transports the tape back to the PDP room, logs in to the PDP-11/70, and performs a TPI. Below follows the sequence of events. In this example, the user retrieves tape #90017 from the PDP-11 room and the NASC operators tell him to use U90344 as the serial number.

```
//M9033ASM JOB (1051601A,R332,90,9,300,,1,N),TAMBERRINC,
//   NOTIFY=M9033
//*
//* THIS IS M9033.CNTL(ASM)
//*
//*
//* THIS JCL WILL ASSEMBLE A SOURCE PROGRAM AND PLACE THE OBJECT
//* MODULE IN A PDS FOR LATER USE.
//*
//ASSEMBLE EXEC ASMFC,
//   PARM='XREF(FULL),NODECK,OBJECT'
//SYSGO   DD   DSN=M9033.SCOPE.OBJ(SBSTRC)
//SYSIN   DD   DISP=SHR,DSN=M9033.SCOPE.SBSTRC.ASM
```

FIGURE 4-01: Sample JCL for Assembling String Handling Routine

```
//M9033CP1 JOB (1092000J,PRC,120,20,200),TAMBERRINC,
// NOTIFY=M9033,MSGCLASS=Q
//COMP EXEC FORTXCL,
//  PARM.FORT='XREF,NOLIST,OPT(1),MAP,LINECOUNT(75)',
//  PARM.LKED='XREF,LET,LIST,MAP',FORTRGN=192K
//FORT.SYSLIN DD DSN=M9033.SCOPE.OBJ(MAIN),DISP=OLD
//FORT.SYSIN DD DSN=M9033.SCOPE.MAIN.FORT,DISP=SHR
//LKED.SYSLIN DD DSN=M9033.SCOPE.OBJ(MAIN),DISP=SHR
//LKED.SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//              DD DSN=SYS4.TEKTRONX,DISP=SHR
//LKED.SYSLMOD DD DSN=SYS9.DCAFORT.LINKLIB(SCOPE),DISP=SHR,
//  UNIT=3330
//LKED.LIB DD DSN=M9033.SCOPE.OBJ,DISP=SHR
//LKED.SYSIN DD *
    INCLUDE LIB(ASK)
    INCLUDE LIB(BLANKR)
    INCLUDE LIB(BOX)
    INCLUDE LIB(CIRCLE)
    INCLUDE LIB(CALTEK)
    INCLUDE LIB(DECODE)
    INCLUDE LIB(DEFALT)
    INCLUDE LIB(DRAW)
    INCLUDE LIB(FETCH)
    INCLUDE LIB(HELP)
    INCLUDE LIB(LLSQ)
    INCLUDE LIB(PRMSG)
    INCLUDE LIB(SPECL)
    INCLUDE LIB(SPLINE)
    INCLUDE LIB(SYMHGT)
    INCLUDE LIB(STNDRD)
    INCLUDE LIB(TRANS)
    INCLUDE LIB(WINDOW)
    INCLUDE LIB(GSTFF)
    INCLUDE LIB(GSTPL)
    INCLUDE LIB(TRNSLT)
    INCLUDE LIB(SBSTRC)
    INCLUDE LIB(SBSTRI)
    INCLUDE LIB(INDEX)
    INCLUDE LIB(VERIFY)
    INCLUDE LIB(CONCAT)
    NAME SCOPE(R)
```

FIGURE 4-02: Sample JCL to Compile and Link Edit a Module of the CALCOMP Preview Program

NASC

1. LOGON to NASC computer.
2. Do a LBGET by typing in:

   LBG CLTKPDP DS('R831.LIBRARY') FORT RON
3. The result is a data set called

   Userid.CLTKPDP.FORT. Now, perform an ITP, by typing in:

   ITP CLTKPDP.FORT your name VOL(U90344) TIME(20)

When the job submitted by the ITP is finished the user retrieves his tape and transports it to the PDP room. He types in the following commands on the PDP-11/70.

PDP-11

1. LOGIN to the system.
2. Mounts tape by typing in:

   ALLOC MM0:

   MOUNT/FOREIGN MM0: 90017
3. Performs a TPI by typing in:

   MCR TPI CLTOTK.FTN

This results in creating a data set in your UIC called CLTOTK.FTN.

4.2 Extensions to a Different Machine. If the CALCOMP Preview System is to be implemented on a different machine, such as Honeywell, CDC, or Burroughs, several changes to the source code may be required to allow for differences in machine operation. These changes will probably involve the areas described below:

a. Size of the CALCOMP records - On the IBM and the PDP-11, CALCOMP records are 360 characters. On another machine, the record size may well be different from either the IBM or the PDP-11. Affected routine: FETCH.

b. Different internal representation - Conversion to another machine will require changes to any source code that converts internal machine representation of a character to a number. Affected routines: STNDRD, FETCH, TRNSLT.

c. Baud rate - The baud rate of the Tektronix terminal connected to a different computer may be different than either the PDP-11 or the IBM. (Thus, parameters in the calls to INITT and TERM may need to be changed.) Affected routine: BLANKR.

d. Integer Representation - On the IBM, the default integer is a full word integer. On the PDP-11, the default integer is a half word integer. On both machines, the INT function converts a real number to a default integer. Because of overflow, real numbers must be converted to full word integers. Hence, the PDP-11 and IBM required different functions for real to full word integer conversion. When converting to another machine, be sure real to integer conversion results in a full word integer. Affected routines: CALTEK, STNDRD, SPECL.

e. String Manipulations - The routines to manipulate strings are machine dependent. On the IBM, they are written in assembly language. On the PDP-11, they are written in PDP-11 Fortran 4 Plus. These routines will need to be rewritten for a different machine. Affected routines: GSTRE, GSTRI, CONCAT, INDEX, SBSTRI, SBSTRO, TRNSLT, VERIFY.

f. File Name Specification - On the PDP-11, the input file name followed a strict convention; hence the need for subroutine GETFIL called by subroutine FETCH. If conversion to another machine requires an input file of a certain naming convention, GETFIL needs to be changed. Otherwise, GETFIL may not be needed. Affected routines: FETCH, GETFIL.

g. Messages - Instructions on how to use certain commands such as "DUMP" and "PLOT" may need to be changed because the naming conventions for files. Affected routine: HELP

h. Exclamation Point - On some machines, such as the IBM, the exclamation point which delimits strings is unprintable and may not be interpreted correctly when transferring to another machine. Thus, the exclamation point needs to be explicitly specified. Affected routines: DECODE, FETCH.

The IBM version of the CALCOMP Preview System, as implemented on the NASC AS/5-3 at DCEC, can be run via TSO with no overlaying. Due to the size of the program, the PDP-11 version needed an overlay structure to permit successful linkage edit. See Figure 4-03 for the listing of the overlay structure on the PDP-11/70. A similar overlay structure may need to be built when trying to run on another machine.

4.3 Additional Commands. If a new command is to be added to the CALCOMP Preview System, changes will need to be made to modules HELP and MAIN as well as writing any software pertinent to the new command. In HELP, the prompt telling the user all available commands will need to be changed. Also, a message telling how to use the new command will need to be added to HELP. Another statement number will be added to the computed GO TO to branch to the new message.

In module MAIN, IF statements will be needed to implement the new command. One IF statement is needed to implement the actual command while another IF statement is needed for the call to HELP with the message number to be one greater than the current highest message number.

A new key word is added to the DATA statements to correspond to the new command. Currently, variables compared to the key words allow for one, three, four, five, six, seven, eight, and ten character responses. REPLY5? These variables are named REPLY2, REPLY4, REPLY6, REPLY7, REPLY8, REPLY9, and REPL11, respectively, to allow for the actual length of the response followed by one space. Thus a three character response such as "BOX"

```
         .ROOT   CLROOT-*(BLANKR,BOX,HELP,WINDOW-ASK,DRAWSG)
CLROOT:  .FCTR   MAIN-DEFALT-FETCH-GETFIL-LLSQ-TRANS-OPMSG-BLOCK-STRRTN
STRRTN:  .FCTR   GSTRL-GSTRE-VERIFY-CONCAT-SBSTRI-SBSTRO-INDEX-TRNSLT
DRAWSG:  .FCTR   DRAW-DECODE-CALTEK-SYMHGT-*(DRSUB)
DRSUB:   .FCTR   WRAPUP,CIRCLE,STNDRD,SPECL,SPLINE
         .END
```

FIGURE 4-03: Listing of Overlay Structure on PDP-11/70
([60,20]CLTOTK.ODL)

is compared to REPLY4, which equals "BOX". Currently, the REPLY variables provide only for responses currently in the program. Hence, no REPLY3, REPLY5, or REPL10. If the new command is nine characters in length, a variable named REPL10 should be declared for use with the compares used to interpret user responses. Likewise, if a new command is two characters in length, a variable named REPLY3 should be added.

APPENDIX A

Mathematical Background

(Note: This appendix is intended for a reader who has a knowledge of mathematics through elementary linear algebra).

The purpose of this appendix is to present the mathematical background for the computations which the CALCOMP Preview System performs in simulating the CALCOMP plotter. Anyone who modifies the methods used to compute plotting coordinates must have a good understanding of this Appendix.

I. Coordinate Systems used:

a. CALCOMP plotter: the plotter draws on a rectangular bed which is 48 inches high by 82 inches wide. The origin of coordinates is at the lower left hand corner, and the unit of measurement is 1/5080 of an inch. Normally, therefore, the largest Y value is 48 X 5080 = 243840, and the largest X value is 82 X 5080 = 416560. The command system of the plotter provides for a mechanical or mathematical relocation of the origin to any arbitrary position, however.

b. TEKTRONIX graphic terminal, model 4014-1: The origin of coordinates is at the lower left hand corner of the plot area. The plottable points are at integer coordinates. The largest Y value is 3120, and the largest X value is 4095.

II. Coordinate Transformation Technique used.

If the coordinates of the CALCOMP system are considered to be the C-plane and those of the Tektronix are considered to be the T-plane, the basic problem facing the program is to define, and then apply, a rigid transformation from the C-plane into the T-plane that will possess desired characteristics, normally expressed in terms of scaling, offset, and rotation, or something equivalent.

The rigid transformation may be expressed as a matrix A with the following properties:

If $\vec{C} = (xc,yc)'$ is a vector of coordinates in the C-plane, and $\vec{T} = (xt,yt)'$ is the corresponding vector of coordinates in the T-plane, then the transformation is rigid if

$$A \begin{pmatrix} xc \\ yc \\ 1 \end{pmatrix} = \begin{pmatrix} xt \\ yt \end{pmatrix}$$ and the submatrix

$$A1 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

is a multiple of an arthogonal matrix (and hence nonsingular). Clearly A must be a 2 X 3 matrix; the vector

$$\vec{o} = \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix}$$ is the point to which the origin of

coordinates in the C-plane is mapped.

Alternatively, one could write the transformation as

$\vec{T} = AC + \vec{o}$ and then solve for

C, given $\vec{T}$, by

$C = A^{-1}(\vec{T} - \vec{o})$

The program governs the transformation by establishing three pairs of noncollinear points which correspond to each other in the C-plane and the T-plane, and using the relationship to determine the matrix A, as follows.

If $\vec{C_j}$ and $\vec{T_j}$ $(j = 1,2,3)$

are corresponding points, then A must satisfy

$$\begin{pmatrix} \vec{C_1}' & 1 \\ \vec{C_2}' & 1 \\ \vec{C_3}' & 1 \end{pmatrix} A' = \begin{pmatrix} \vec{T}' \\ \vec{T_2}' \\ \vec{T_3}' \end{pmatrix}$$

Within the program, the subroutine LLSQ is used to solve this matrix equation. The matrix A is stored in the array CLTTK (3,2) in labeled COMMON.

The three non-collinear points are the lower left, upper left, and upper right corners of the area to be plotted. Their coordinates are stored in the arrays CLCRNR (3,2) ('CALCOMP-Corners') and TKCRNR (3,2) ('TEKTRONX-Corners') respectively, in labeled COMMON. Assignment of these values is done by various subroutines in the program, depending on the function that the user selects.

III. Spline interpolation.

The CALCOMP system performs spline interpolation by using four points, say X1, X2, X3, and X4. Given the four points, the system draws a smooth curve from X2 to X3, using X1 and X4 to govern the directions at the beginning and end of the curve. After this, the old X1 is discarded, X2 becomes X1, X3 becomes X2, X4 becomes X3. If a new X4 is supplied, the process is repeated.

To start and end the plot of a curve, it is necessary to supply "phantom points" which are never plotted, but which serve as the first X1 and the last X4. If the first X1 is not distinct from the first X2, or the last X4 is the same as the last X3, the system still works, however.

This program uses a pair of cubic equations in an auxiliary variable, or parameter, to approximate this function. The cooefficients are determined in a manner similar to that used for coordinate transformation.

The auxiliary variable t is chosen so that the points map thus:

T	X
-3	X1
-1	X2
1	X3
3	X4

Then the coordinates are computed by

$$x = a3t^3 + a2t^2 + a1t + ao$$

$$y = b3t^3 + b2t^2 + b1t + bo$$

with t ranging from -1 to 1 with an increment just large enough to make a change of one unit on the CRT face. The coefficient vectors

$\vec{A} = (a3\ b2\ a1\ ao)'$

$\vec{B} = (b3\ b2\ b1\ bo)'$ are determined by

$$\begin{pmatrix} -27 & 9 & -3 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 27 & 9 & 3 & 1 \end{pmatrix} \begin{pmatrix} \vec{A} & \vec{B} \end{pmatrix} = \begin{pmatrix} X1 & Y1 \\ X2 & Y2 \\ X3 & Y3 \\ X4 & Y4 \end{pmatrix}$$

Again, the subroutine LLSQ is used to solve this system and determine the coefficients.

If either end point is not distinct, the degree of the parameter polynomial is reduced to two, and appropriate changes are made in the matrices.

APPENDIX B

This appendix contains a cross reference table between the variables contained in the common block COMBLK and the programs comprising the CALCOMP Preview System. Any entry in the table containing a "U" means that the variable is used within a program but not changed. Any entry containing a "C" means the variable is changed.

	ASK	BLANKR	BOX	CALTEK	CIRCLE	DECODE	DEFALT	DRAW	WRAPUP	FETCH	HELP	LLSQ	MAIN	OPMSG	SPECL	SPLINE	STNDRD	SYMHGT	TRANS	WINDOW	GETFIL
CLCRNR		C	C				C		C				C							C	
CHCRNR		U					C		C				C							C	
TKCRNR		C	C				C		C					U						C	
SCCRNR		U	U				U		U				C							U	
BDCRNR							U		U				C							U	
CHURTC							U						C								
CLTTK		C	C	U	U			C						C							
PHI			U										C								
TANPHI			U										C								
COTPHI			U										C								
SINPHI			U										C								
CSCPHI			U										C								
COSPHI			U										C								
SECPHI			U										C								
PSI		U					C						C							C	
COSPSI		U					C						C								
SINPSI		U					C						C							C	
PSIX		C																			
SINPSX		C	U																		
COSPSX		C	U																		
AVALUE						C	C														
BVALUE						C	C														
EVALUE						C									U		U				
FVALUE						C									U		U				
IVALUE					U	C															
JVALUE					U	C															
NVALUE						C									U		U				
PVALUE				U	U	C	C														
QVALUE				U	U	C	C														
RVALUE				U	U	C	C														
SVALUE				U	U	C	C														
UVALUE				U	U	C	C														
VVALUE				U	U	C	C														

	ASK	BLANKR	BOX	CALTEK	CIRCLE	DECODE	DEFALT	DRAW	WRAPUP	FETCH	HELP	LLSQ	MAIN	OPMSG	SPECL	SPLINE	STNDRD	SYMHGT	TRANS	WINDOW	GETFIL
SNTNCE						C				C			U								
SPLTXT										C				C			C				
LETTER						U															
GVALUE				U	U	C	C	C													
LVALUE						C								U							
MVALUE						C		C													
TVALUE						C															
HEIGHT															C		C				
INFILE										U											C
LENSNT						U				C											
NSNTNC						C															
SPCIL						U				C					U						
SPLLEN										C				U							
XMAX		C		C					C												
XMIN		C		C					U												
YMAX		C		C					C												
YMIN		C		C					U												
XOFST		C	C					C					C								
YOFST		C	C					C					C								
XORG			U					C													
YORG			U					C													
XCAL				U	C	C		C							C		C				
YCAL				U	C	C		C							C		C				
XTEK		C		C	U			U						U	U	U	U				
YTEK		C		C	U			U						U	U	U	U				
XBFORE				C	U			U													
YBFORE				C	U			U													
OLDX		C		C				U													
OLDY		C		C				U													
PREVX		C		C																	
PREVY		C		C																	

	ASK	BLANKR	BOX	CALTEK	CIRCLE	DECODE	DEFALT	DRAW	WRAPUP	FETCH *	HELP	LLSQ	MAIN	OPMSG	SPECL	SPLINE	STNDRD	SYMHGT	TRANS	WINDOW	GETFIL
CURVE								U								C					
DFLTON		U					C													C	
EXACT					U								C			U					
FILOPN										C			U								C
FRAME		U											C								
HALT								U													
PNDOWN				U		C		U													
PLOTNG		C						C	C	C			C								
RRIGIN								U					C								
STEP				C	C	C		U					C								
ENDFIL										C			C								
RRGIND		C	U					C													

* In subroutine FETCH, the bit FILOPN is used only in the NASC version. On the PDP-11/70, FETCH calls GETFIL instead.

Note: LLSQ does not access COMMON, but passes all values as arguments.

APPENDIX C

Differences Between CALCOMP Routines on IBM and on PDP-11

1. String manipulation routines in IBM are written in assembly language. String manipulation routines on the PDP-11 are written in PDP Fortran 4 PLUS.

   Affected routines:

GSTRL	SBSTRI
GSTRE	SBSTRO
INDEX	CONCAT
VERIFY	TRNSLT

2. In the subroutine BLANKR, the subroutines INITT and TERM require different parameters due to differences in baud rate and terminal characteristics.

   a) IBM calls to INITT and TERM are:

      CALL INITT (180)
      CALL TERM (2,4096)

   b) PDP-11 calls to INITT and TERM are:

      CALL INITT (1100)
      CALL TERM (3,4096)

3. In the subroutine CALTEK, different built-in functions are used to convert a real number into an integer. The IBM version uses the INT function. The PDP-11 version uses the JINT function.

4. In the subroutine DECODE, the ASCII character (!) is not available on the EBCDIC character set used in the NASC system. The corresponding character used is the right bracket (]). The exclamation point (!) is used on the PDP-11.

5. In the subroutine FETCH, the variable SPCIL, which contains a pointer to a designated special character is computed differently due to differences in internal representation on the IBM and PDP-11. On the IBM, SPCIL is the quotient of ISPCL (the machine representation of the special character) and the constant 16777216. On the PDP-11, SPCIL is equal to the difference between ISPCL and the constant 538976256.

Due to the naming convention of PDP-11 Fortran files, the PDP-11 version of FETCH calls a subroutine GETFIL with a parameter 0 or 1, while the IBM version merely sets a bit FILOPN true or false.

The exclamation point character is not available on the NASC system, and is handled as described in para 4, above.

6. In the subroutine SPECL, different built-in functions are used to convert a real number into an integer. The IBM version uses the INT function. The PDP-11 version uses the JINT function.

7. In the subroutine STNDRD:
   a) Due to different internal representations in the IBM and PDP-11, the variable PTR, which is used as a pointer, is calculated differently. In the IBM, PTR is the quotient of K1UP(1) divided by 256. In the PDP-11, PTR is the difference between K1UP(1) and 8192.

   b) Different built-in functions are used to convert a real number into an integer. The IBM version uses the INT function. The PDP-11 version uses the JINT function.

8. In the subroutine HELP, the messages for the PLOT and DUMP commands are different in accordance with differences between the PDP-11 and IBM.

9. On the NASC, DRAW and WRAPUP are compiled simultaneously. On the PDP-11, they are stored in separate object modules.

APPENDIX D

Structure of the CALCOMP Plot File

The structure of the CALCOMP plot file, the primary input to the CPS, is governed by the design of the CALCOMP plotter, as implemented by the software distributed by the CALCOMP manufacturer. This structure, in turn, governed the design of subroutine FETCH, which performs the input function for the program.

The logical structure of the files is the same in any implementation of the CALCOMP plotter; however, the physical structure of the files differs greatly among the various computers that support it, such as the PDP-11, NASC AS-5/3, or IBM 370 systems. This appendix describes only the logical structure of the records in the files. Any implementation of this program must rely on the operating system for the physical input function.

The file is composed of a sequence of sentences, which provide commands and data. The sentence, in turn, is composed of a sequence of words, which carry the individual data items. Each word begins with a letter, which may be followed either by numeric data or by a character string. A character string is a non-null string of characters preceded and followed by an exclamation point. The exclamation point character may be transmitted only as a single character, in a group of three exclamation points. In that case, the first and third exclamation points are used as delimiters, and the second one is the data. Any characters contained in a character string are treated as data, and lose any significance they might have had otherwise. The sentence is ended by a period not contained in a character string.

The records in the file start with a four-byte constant string called a sync code, which serves to mark the beginning of the words in the record. The words in the record are contained between the sync code and a dollar sign which is outside a character string. The delimiting dollar sign, and any data that follows it, is not otherwise significant.

Sentences may be continued over more than one logical record, but they are always broken between words, so that logical records contain complete words, but not necessarily complete sentences.

The data is transmitted in modified ASCII8, in which the space character is binary zero, and all other characters follow in the ASCII character collating sequence, up to the underscore character, which has binary value 63.

The following are examples of CALCOMP sentences, and their interpretation. A complete description is given in reference 1.2b.

a. N17G1D2X-66040Y86360.

   The parts are:

   N17 - identifies and numbers the sentence.

   G1 - move or draw to the following coordinates.

   D2 - raise the pen - i.e. move, do not draw.

   X-66040 - X coordinate is minus 66040 "CALCOMP Units", which normally are scaled 5080 to one inch.

   Y86360 - Y coordinate is plus 86360 CALCOMP units.

   . - marks the end of the sentence.

b. N3T1G25XYM1.

   The parts are:

   N3 - identifies and numbers the sentence.

   T1 - search address for operator use.

   G25 - reorigin the coordinate system to the present location of the pen.

   X - new X coordinate is 0.

   Y - new Y coordinate is 0.

   M1 - perform a temporary halt.

   . - marks the end of the sentence.

c. N145G52!HAPPY FACE!.

The parts are:

N145 - identifies and numbers the sentence.

G52 - command to print a string of characters.

!HAPPY FACE! - The string of characters to be printed. The exclamation points are delimiters.

. - marks the end of the sentence.

Figure D-01 contains a listing, in EBCDIC characters, of a plot file used as test and demonstration data, which was produced by the TDUMP subroutine provided by CALCOMP and described in Reference 1.2b. The listing, as produced by TDUMP, differs slightly from the true content of the file, as follows:

The sync code, which is the first four characters of each logical record, is represented as '????'. The ASCII value is '?99?'.

The delimiters on quoted strings are shown as single quote marks. They are, in fact, exclamation points, ASCII8 character 33 (Hex 21). This character is represented as ']' on one of the EBCDIC print sets used on the HSF NASC AS/5-3 computers, and is unprintable on the other.

Note that when the data in a record is terminated by a data delimiter ($), the characters following the delimiter are left unchanged, and are not significant to the program.

The length of the logical record is determined by parameters set within the CALCOMP subroutines used at the installation. At the HSF, the subroutine that governs the length is called BUFF; the governing parameter is JMAX, which sets the record at 90 words, or 360 characters. Of this, four are used for the sync code, and 356 remain for data.

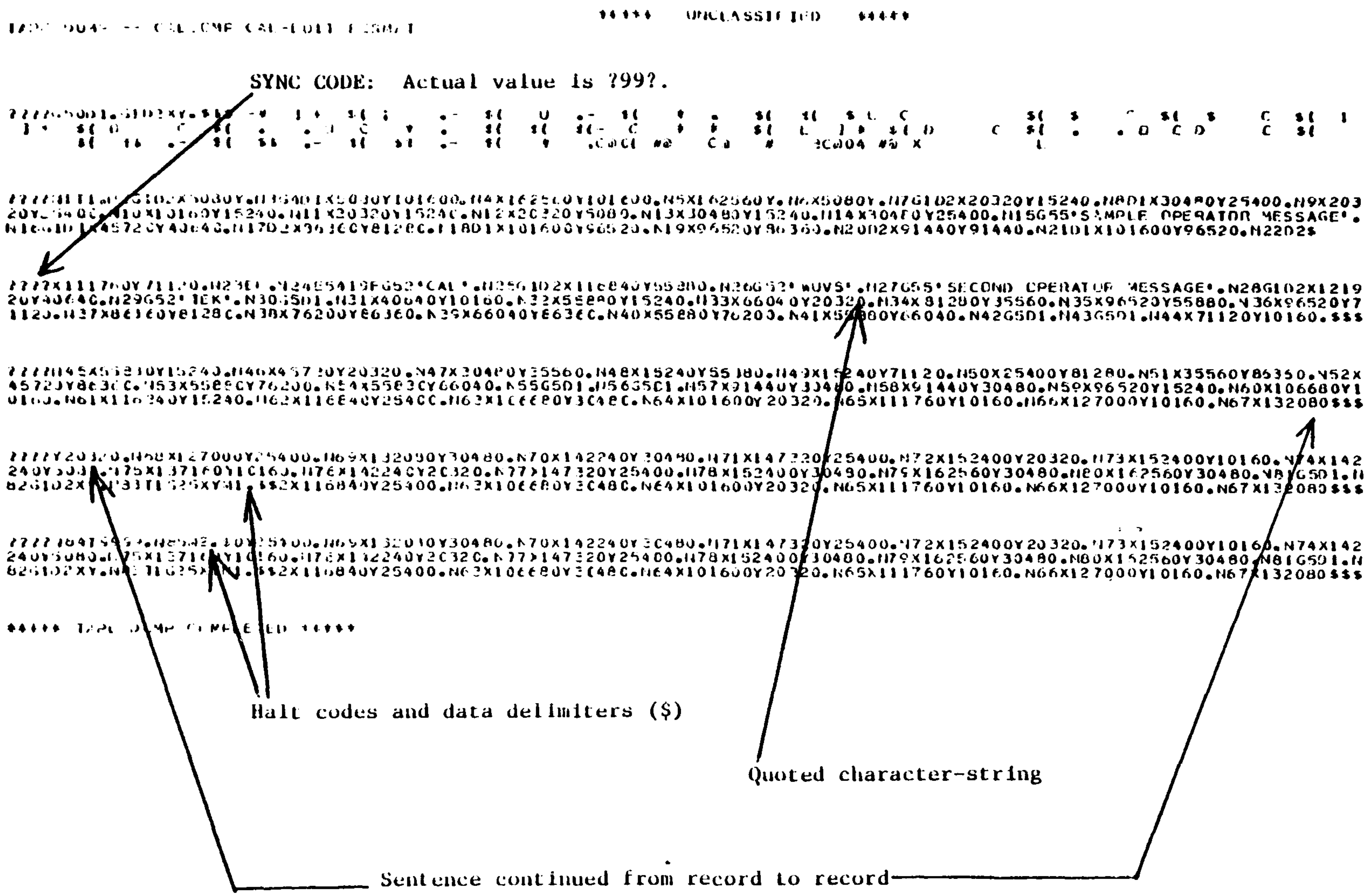


FIGURE D-01: Sample CALCOMP Plot File

DATE
FILMED

-8